Hands-on with ASP.NET MVC

Covering MVC 6

Rahul Sahay

Hands-on with ASP.NET MVC

Covering MVC 6

Rahul Sahay

Hands on With ASP.NET MVC

Hands on With ASP.NET MVC

Rahul Sahay

Quills Ink
Publishing

Copyright © 2014 Rahul Sahay

Hardcover ISBN : 978-93-84318-52-9
Softcover ISBN : 978-93-84318-53-6
Ebook ISBN : 978-93-84318-54-3

Design and Setting By
Quills Ink Publishing
www.quillsink.com

All rights reserved. No part of this publication may be

reproduced, distributed, or transmitted in any form or by means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the publisher.

The views and characters expressed in the book are of the author and his/her imagination and do not represent the views of the Publisher.

Contents

Chapter 1: Getting Started

WHAT DO you find in this CHAPTER?

Introduction

Understanding ASP.NET MVC

The MVC Pattern

Brief History of MVC Pattern

MVC Life Cycle

Software Requirements

Application Glimpse

Creating a MVC application

How MVC applications are

structured
Summary
Chapter 2: Understanding Controllers
WHAT DO you find in this CHAPTER?
Introduction
Controllers
Working with your 1st controller
Action Results
Action Selectors
Action Filters
Summary
Chapter 3: Views
WHAT DO you find in this CHAPTER?
Introduction
Views

Syntax of Razor Views
How to Prevent Cross Site Scripting Attack
More on Razor Syntax
Layout Views
HTML Helpers
HTML.ValidationSummary
Some more HTML Helpers
Partial Views
Viewbag and Viewdata
Summary
Chapter 4: Getting Started with Data modeling
WHAT DO you find in this CHAPTER?
Introduction
Using Entity Framework
Entities Creation

Database Migrations
LINQ
Search Query
Summary
Chapter 5: Data modeling- Part 2
WHAT DO you find in this CHAPTER?
Introduction
Organizing Movie Controller
Reviews Listing
Create and Edit Facility of Review
Validation Annotations
Summary
Chapter 6: JavaScript and AJAX in MVC
WHAT DO you find in this CHAPTER

Introduction
Managing Scripts
Ajax Helpers
Async Search
Paging
Summary
Chapter 7: Security with ASP.Net MVC
WHAT DO you find in this CHAPTER?
Introduction
Authentication
Membership and Forms Authentication
Authorize
Seeding Membership
Cross Site Vulnerability
OpenID and OAuth

Summary
Chapter 8: Caching - ASP.Net Terminology
WHAT DO you find in this CHAPTER?
Introduction
Caching
Cache Profiles
Localization
Resource Files
Diagnostics
ELMAH
Summary
Chapter 9: Unit Testing
WHAT DO you find in this CHAPTER?
Introduction
TDD

Home Controller Tests
Web Tests
Load Tests
Code Coverage
Summary
Chapter 10: Deployment
WHAT DO you find in this CHAPTER?
Introduction
Configuration Files
Hosting ASP.Net MVC Application
Deployment Preparation
Deployment on IIS
Deployment to Azure
Summary
Chapter 11: Routing Basics
WHAT DO you find in this

CHAPTER?
Introduction
Route Definition
Areas
Ignore Routes
Debugging Routes
Constructing URLs from Route
Route Data
Route Constraints
Summary
Chapter 12: Web API and Dependency Injection
WHAT DO you find in this CHAPTER?
Introduction
Why Web API
Creating a new empty Project
Adding Web API to an ASP.Net

Project
Creating your 1st controller
Dependency Injection
Implementing POST
Implementing DELETE
Implementing PUT
Summary
Chapter 13: Making a Mobile Site Using Jquery Mobile
WHAT DO you find in this CHAPTER?
Introduction
Setup
Viewports, Widgets and Attributes
Themes and Swatches
Media Query
Layouts and Pages

Collapsibles
Grids and Buttons
Forms
Summary
Chapter 14: ASP.NET MVC
WHAT DO you find in this CHAPTER?
Introduction
Scaffolding
ASP.Net Identity
One ASP.Net
Bootstrap
Attribute Routing
Filter Overrides
Summary
Chapter 15: ASP.NET MVC
WHAT DO you find in this CHAPTER

Introduction
Features
Prerequisites
Setting up VM
Getting Started with Visual Studio 14
Creating your 1st Visual Studio 14 MVC Project
Configurations in MVC Project
project.json
Startup class
Adding a Class Library Project
Deployment to Azure
Bin
Self Hosting
Summary
Index

Introduction

Hands-on with ASP.NET MVC book give you a fine Launchpad to get started on MVC. I have written the book in such a fashion that you will solve one case study while studying the book. By the time you will finish the book, you would have completed web version and mobile version of development using MVC. Now, one important point to understand here that ASP.NET MVC is a major shift for all Microsoft developers. ASP.NET MVC emphasizes clean architecture, design patterns and testability. At every part of the book you will get practical

example of all the theoretical concepts with complete demo.

ASP.NET MVC is basically a web development framework from Microsoft which combines the features of MVC (Model-View-Controller) architecture. Now, ASP.NET MVC is not something which is built from the ground zero. It built on the top of ASP.NET, so developers enjoy almost all the ASP.NET features while building the MVC application. ASP.NET MVC is an alternative approach to an existing Web Forms approach.

This book is basically comprised of 15 chapters covering all the parts of MVC

development. 1st chapter starts with the introduction of ASP.NET MVC where in you will see how MVC gained maturity in different releases. 1st chapter will also showcase that what you are going to build by giving the application glimpse in its finished form. Then, you will start creating your MVC project right from the scratch. In 2nd Chapter you will get the complete idea of controllers. Here, after understanding the basics of controllers, you will be writing your 1st controller. 2nd chapter also covers different types of action results, filters and many more things. 3rd chapter will have view introduction with syntactic sugar. This section also explains some of the core

concepts of data — (dash) attributes that how views behind the scenes emit these tags. Apart from HTML helpers, you will also work with different types of view in it. Chapter 4 and 5 will be all about data modeling. In this chapter you will come to know how to get started with entity framework, also how to setup the entities and migrations. Here you will also see different types of validations in action.

Chapter 6 is all about Client side scripts. Its main focus would be on how to manage scripts, how to work with Ajax helpers and how to write an Async Search. Chapter 7 is covering all the security related features like

Membership, Authentication, Roles, Open Authentication and many more things. Now, chapter 8 is all about infrastructure topic like Caching and different Diagnostics techniques. This topic stresses how to improve the Application performance by implementing caching. 9th chapter is all about testing. It start with unit testing where in you can test your application logic, then web testing and load testing where in you can test your website performance in different test scenarios. Then we have 10th chapter which is all about deployment. In this case you will 1st deploy the application to IIS, then on internet on AZURE. Chapter 11 is all about routing concepts. This chapter

deals with different routing rules. Then, chapter 12 comprise of Web API and Dependency Injection. Both of these topics are really important and interesting as well. You will learn how to implement both with a complete application demo. In chapter 13, you will be optimizing your application in such a way that it will support mobile browsers as well. Then, Chapter 14 is dedicated to the new features of ASP.NET MVC 5 with complete demo. Last but not the least which is the bonus part of this book MVC 6; here, in chapter 15 you will start from the features discussion, then prerequisites required for MVC 6. MVC 6 is a major breakthrough from Microsoft side

towards being Open Source Compliant. So, stay tuned for a stunning journey.

Acknowledgments

This book would not have been possible without the patient and loving support of my wife Nivi, who had to take over much of the household responsibility apart from her teaching activities, so I could whisper to myself in the corner and do my activities with my laptop. Many thanks also go to my publisher, Rohan Vij, who took this responsibility, to publish the book in a precise and correct way.

Appreciation also goes out to my Mom and Dad for believing in me and always keeps on encouraging me to complete the book in a best possible way. Also, I need to acknowledge you, the reader, for your desire to continue your own growth not only as developer but a smart developer. Obviously, your efforts to improve your skills make me all work to improve myself to keep up the good work. Kudos to you, and keep setting the pace at your workplace!

I would also like to Thank Sanjay for writing the ***FOREWORD*** for me. He took his time from his day to day busy schedule, read the complete book; gave his suggestions for improvement and

finally after implementing his suggestions he wrote foreword for me.

WHO SHOULD TAKE THIS BOOK

Hand-on with ASP.Net MVC is designed to teach ASP.NET MVC, from grass root level to advanced topics. I have written the book in such a way that even if you are new to ASP.Net background, you won't feel difficulty in understanding the concepts.

I also understand that many of our readers are familiar with ASP.NET Web

Forms, so one important point to understand here that ASP.Net MVC is not a replacement to Web Forms at all. Now, onwards there will be one more design technique to build websites.

I have worked hard to make sure that this book is valuable for developers who are experienced with ASP.NET MVC, or even freshers coming out of college. This book is also very well suited to 3rd year and Final year student as this will help them greatly while building their projects. Also, since ASP.NET MVC is one of the hottest trends across industry, so from the point of interview preparation as well, it will be a good reference book. Now, apart from MVC

you will also learn how to convert the same app to support mobile browsers, also how to host the app on Azure (Microsoft's Cloud Version).

Foreword

Today I am really thrilled to introduce this book covering the latest release of ASP.NET MVC, written by one my friend Rahul Sahay.

Rahul was developer when I interacted with him for the first time. He is tech extrovert and interacting with wide variety of technology to gain knowledge, guideline and feedback. His interactions with the diverse ASP.NET community give him some great insights on how

developers can begin, learn, and master ASP.NET MVC. He is helping number of blog followers on various topics which is known to him and generated a lot of fans and followers with his technical skills. With a background rooted in community and open source, I count him not only as an amazing technologist but also a close friend.

There are number of frameworks available in market but ASP.NET MVC 6 brings you a new ASP.NET Web API (a new framework for building HTTP services), updated default project templates that leverage modern web standards, solid mobile web application support, enhanced support for

asynchronous methods, and more with easy integration with other microsoft technology. Currently, MVC 6 is in beta version, which will be officially releasing next year. But, Rahul took this opportunity to include MVC 6 as well in this book. Before touching MVC 6, he has also covered MVC 5 and 4 with complete detail with live demo.

Rahul also developed number of tutorials to help developer understand concepts and also planning to upload a number of videos in line with it. One can always refer his articles on his blog (http://myview.rahulnivi.net/).

Sanjay Kumar Sinha
BTech- IIT
Solution Architect
TCS

Errata

Finally, this book is mostly based on my opinions. I made every possible effort to ensure that there are no errors in the text or in the code. However, no one is perfect, and mistakes do occur. I've spent a lot of time researching the topics in this book and I implement from real world experience as much as possible. With that words, if you find something you believe is absolutely wrong — or even something you strongly disagree with — let me know

@:rahulsahay19@gmail.com.

Chapter 1: Getting Started

WHAT DO you find in this CHAPTER?

⇒ Introduction

⇒ Understanding ASP.NET

MVC

⇒ The MVC Pattern

⇒ Brief History of MVC Pattern

⇒ MVC Life Cycle

⇒ Software Requirements

⇒ Application Glimpse

⇒ Creating a MVC application

⇒ How MVC applications are structured

⇒ Summary

Introduction:-

Hi Friends, My name is Rahul Sahay, your instructor for ASP.NET MVC. In this section I'll walk you through the overview of MVC. I'll also give a brief snapshot of the History of MVC when it's first introduced and then start the development process. I assume that most of the readers who are reading this book are from the ASP.NET background. However, even readers who are new to ASP.NET architecture don't need to bother at all as I start right from the grass root level. So, let's get started.

Understanding ASP.Net

MVC:-

ASP.NET MVC framework basically designed to build the web applications which take the general legacy approach ***Model View Controller*** Pattern. This pattern existed in tech space since decades; this is not something which is being built by Microsoft from ground Zero. However, ***Microsoft*** has introduced its very own MVC framework called as ***ASP.NET MVC Framework***. Also, it's equally important to understand that Microsoft hasn't written this MVC Framework right from the scratch, rather it's been built on the TOP of ASP.NET which means, ASP.NET MVC still has almost all

features of ASP.NET like ***Master Pages, Session Management, Caching***, Different types of Authentication and many more thing.

Also, one more important thing to understand here that ***ASP.NET MVC*** is not the replacement of ***ASP.NET Web Forms***. Microsoft has provided MVC as an alternative approach to build the web application. So, going forward, now there will be two different approaches to build the web application. However, ASP.NET MVC basically follows the same core principle what other languages follow, plus it has got ASP.NET advantage, like it has got the support of ***Managed Execution***

Environment, it uses ***.Net*** feature like ***Extension Methods, DLR*** features and many more. Some of the common features of MVC which I would like to list out are:-

Stateless:- MVC Framework follows the stateless pattern which means there will be no illusion of state at all. Also, there will be no ***Page Life Cycle***, no ***Server Side Controls*** and no ***View State*** also. So, instead of using any server side controls like Grid Controls or Menu Controls, we will be writing and crafting the HTML completely from scratch. So, this way we'll be having complete control on our HTML.

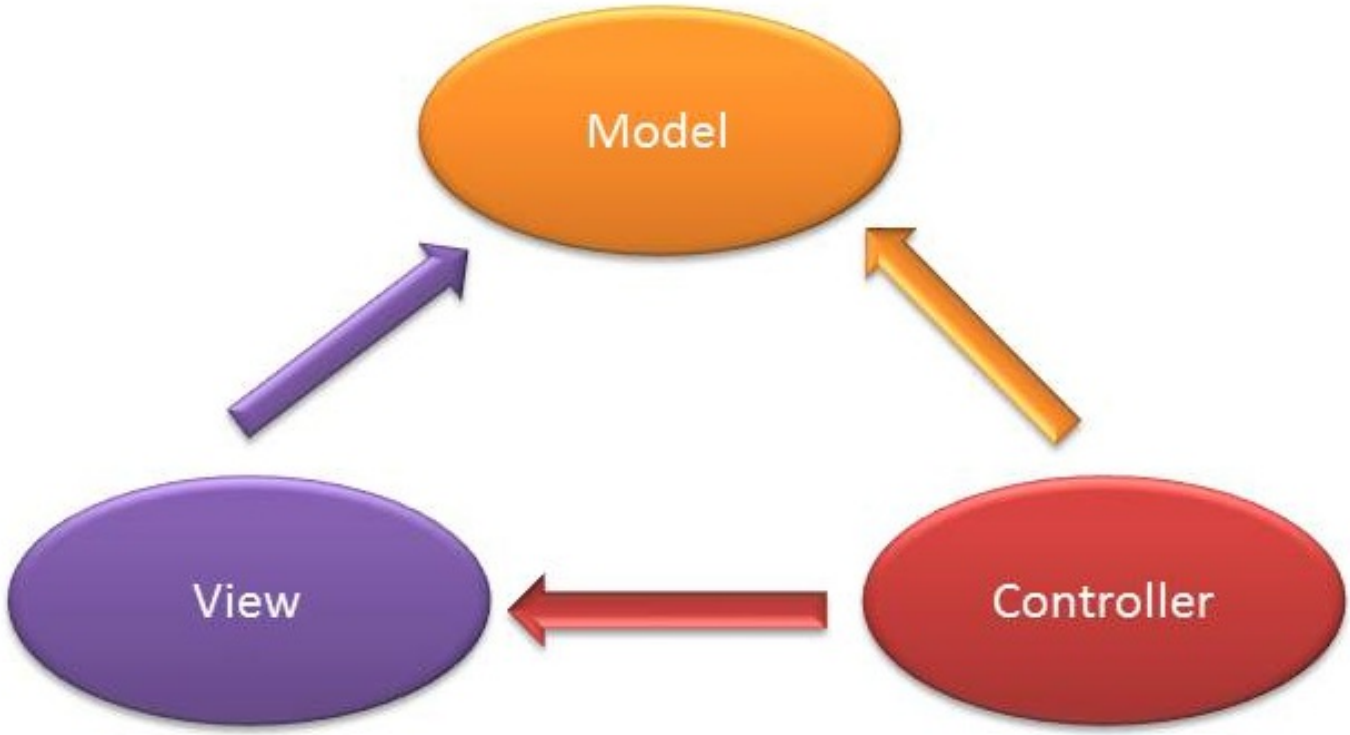
No Page Extensions:- This is also one of the best things which MVC says, so there will be no page extension like ***.aspx***. So, here URL comes is pretty handy and easy to use and easy to remember for readers and also for the search engines. Basically, it is said that pages which are coming without any page extension are ranked higher in the search engine.

Extensible:- This is also one of the common goals for MVC to be completely extensible throughout all the aspects. So, like when we talk about views, one may use aspx view engine to write the view or use the razor to do the same or user can download any other

view engine and use the same with the project as well. But, in this case we'll stick to Razor view engin

Testability:-Testability is the one of the most important factor of using MVC. It's basically quite simple, it maintains complete separation of concerns, and hence it's easy to write and test for each module explicitly. So, here we can easily write test cases for each module explicitly.

The MVC Pattern:-



Having said to my earlier statement that ASP.NET MVC design pattern is not new, this is the technique which is being used since decades across many different technologies like C++, Java, etc. and now in C# as a design pattern for building web applications. It's powerful and stylish design pattern mean for separation of concerns which means within the application itself we'll be

having separate Data Logic and UI Logic. Neither of these are dependent on each other. Hence, each of these can be tested separately by writing unit test cases. Below is the brief snapshot of MVC and its responsibilities.

Model: - Models are basically a collection of classes where in you will be working with data logic as well as business logic. So, basically models are containers which are business domain specific or I would say section which is used to interact with database and also used to manipulate the data to implement the business logic. So, with ASP.NET MVC this is pretty much easy to use data access layer with *Entity Framework,*

ADO.Net or NHibernate combined with custom code.

View: - Views are nothing but a pure HTML which decides how the UI (***User Interface***) is going to look like. All I can say is that views are basically templates which decide the look and feel or presentation of your app. This is the section which is going to present the static HTML with extracted data from database at the desired place.

Controller: - Controllers are basically a bunch of classes which handles communication from the user. This is the one which is responsible for taking incoming HTTP Request and transferring

the control to the appropriate section of the application. So, this is the special class which wires up the relationship between view and model. So, it responds to the user input, interacts with the model and decides which view to render. Basically, Controller classes maintain one convention that it has to be suffixed with the name “controller”.

Brief History of MVC Pattern:-

ASP.NET MVC Pattern is the idea which is being developed by Scott Guthrie in 2007. He is the key architect of this design pattern. As far as my

knowledge goes, I always designate Scott Gu, as the father of .NET. He is the main person who originated the idea of .NET. Since, then Microsoft or Scott's team has added tons of feature in .NET development, which makes developer feel comfortable writing any kind of project using .NET.

ASP.NET MVC 1 Features:-

Here, in this section apart from separation of concerns, Unit tests were added. So, while writing any project, you could write unit test for specific modules side by side. MVC 1st was released on 13th March 2009.

ASP.NET MVC 2 Features:-

ASP.NET MVC 2 was released just in a span of 1 year. This has got some really cool features which make MVC more robust and powerful. This version was released on March 2010. Some of the features of MVC 2 have been listed below:-

- UI Helpers with Scaffolding templates.
- Model validation on Client and Server side.
- Strongly typed HTML Helpers.
- Enhanced Visual Studio tooling.

ASP.NET MVC 3 Features:-

ASP.NET MVC 3 was officially released in 2011. MVC 3 has come with major improvements in many sections. Some of the MVC 3 features are listed below:-

- Inclusion of Razor View Engine.
- .Net 4 Data annotations.
- Robust model binding and validation.
- Inclusion of Global Action Filters.
- Nice JS, JQuery support. Also, included Unobtrusive JavaScript validation.
- JSON Binding
- NuGet Integration to resolve the software dependencies on the fly.

Don't worry I'll do detailing each of the topic in coming chapters; here I just want to make things clean and simple.

ASP.NET MVC 4 Features:-

ASP.NET MVC 4 has done value addition on MVC 3. On the other hand it has become a complete web development package as a whole which offers developer complete suites of development stack. Some of the common features are listed below:-

- ASP.NET Web API
- Improved Project Templates. Added new Ones
- Inclusion of Mobile Projects using JQuery Mobile.

- Various Display Modes
- Asynchronous Controllers
- Bundling and Minification

ASP.NET MVC 5 Features:-

Again ASP.NET MVC 5 has done many value additions to MVC 4. But, all of these changes are around its scaffolding templates, authentication technique, Bootstrap and many more. Some of the common features are listed below.

- Scaffolding
- ASP.Net Identity
- One ASP.Net
- Bootstrap
- Attribute Routing
- Filter Overrides

You will learn complete features with demo in chapter 14.

ASP.NET MVC 6 Features:-

ASP.NET MVC 6 in many terms is a unique framework as it's a major change in MVC. All these features will be discussed in the last chapter in detail, where we'll see the glimpse of MVC 6. Below, I have listed some of the common features of MVC 6.

- Common Framework for MVC, Web API and Web Pages
- Smooth Transiting from Web Pages to MVC
- Built DI First
- Runs on IIS or Self Host

- Based on the new Request Pipeline in ASP.NET vNext
- Runs Cloud Optimized
- No Build Dependency
- Enhanced developer experience
- Open Source
- Cross-Platform Support

Apart from above listed features there are other things also involved which are explained with complete demo in Chapter 15.

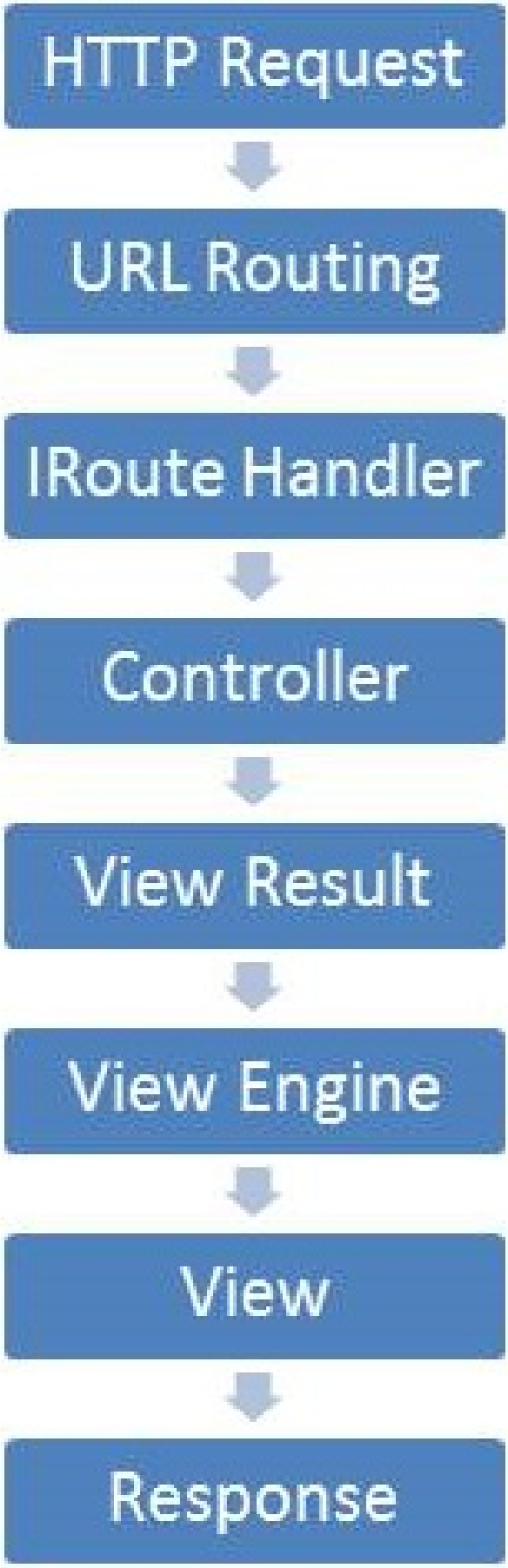
MVC Life Cycle:-

MVC Life Cycle in a nutshell will give you glimpse of overall architecture.

Here, in the below diagram you will come to know How MVC application behaves when it's invoked from the browser. So, as you see in the diagram, as soon as request comes from the browser, it is picked up by the routing engine. Hence, **ASP.NET Routing** is the first step in MVC Life cycle. Basically it is a pattern matching system which matches the request's pattern against the registered patterns in the Route Table. When a matching pattern found in the Route Table, the Routing engine forwards the request to the corresponding **IRouteHandler** for that request. If the routing engine doesn't match the pattern then it returns 404 HTTP Status code.

Then, MVC handler implements IHttpHandler interface and further process the request by using **ProcessRequest** method. Then in third step MVCHandler uses the **IControllerFactory** instance and tries to get a **IController** instance. If successful, then **Execute** method is called. Now, once the controller has been instantiated, Controller's **ActionInvoker** checks which action to invoke on the controller. Next, the action method receives user input, prepares the appropriate response data, and then executes the result by returning a result type. The result type can be **ViewResult**, **RedirectToRouteResult**, **RedirectResult**, **ContentResult**,

JSONResult, **FileResult**, and **EmptyResult**. Now, the next step is the execution of the View Result which involves the selection of the appropriate View Engine to render the View Result. It is basically handled by **IViewEngine** interface of the view engine. Now, at last Action method returns a **string**, **binary file** or a **JSON Formatted** data. The most important Action Result is the **ViewResult**, which renders and returns an HTML page to the browser by using the current view engine.


HTTP Request
URL Routing
IRoute Handler
Controller
View Result
View Engine
View
Response

Software Requirements:-

MVC 4 runs comfortably on any of the following windows Operating Systems.

- Windows 8
- Windows 7
- Windows Vista
- Windows XP
- Windows Server 2008 R2
- Windows Server 2008
- Windows Server 2003

MVC 4 tooling comes by default with Visual Studio 2012/13. This can also be installed on Visual Studio 2010 as well. However, if you are not having the

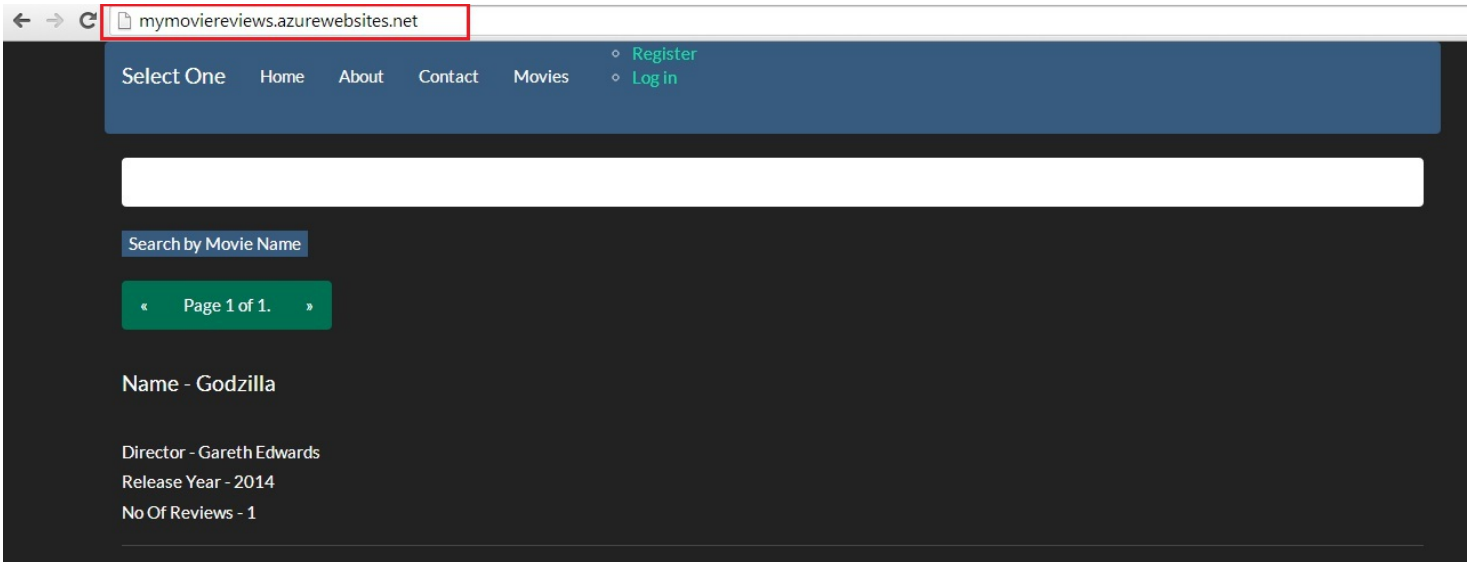
professional version of Visual Studio, you can use free version of visual studio, Visual Web Developer Express from the Microsoft Site itself. If you're using Visual Studio 2010, you can install MVC 4 using the Web Platform Installer

(http://www.microsoft.com/web/gallery/appid=MVC4VS2010).

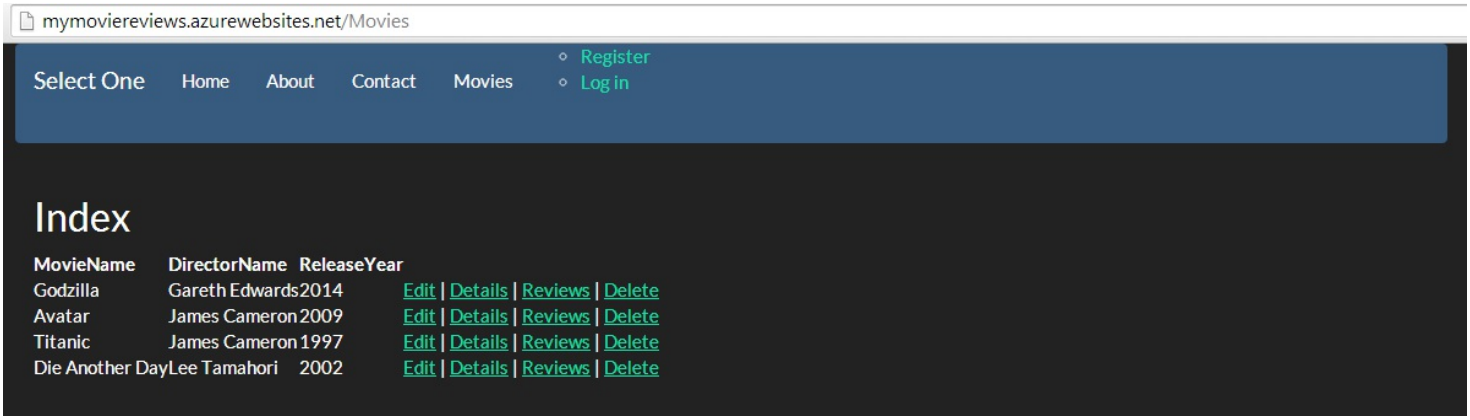
Application Glimpse:-

Now, before jumping to create our 1st MVC application, let me give you demo that what we are going to create here. So, below is the snapshot of "***MovieReview***" site in its finished form.

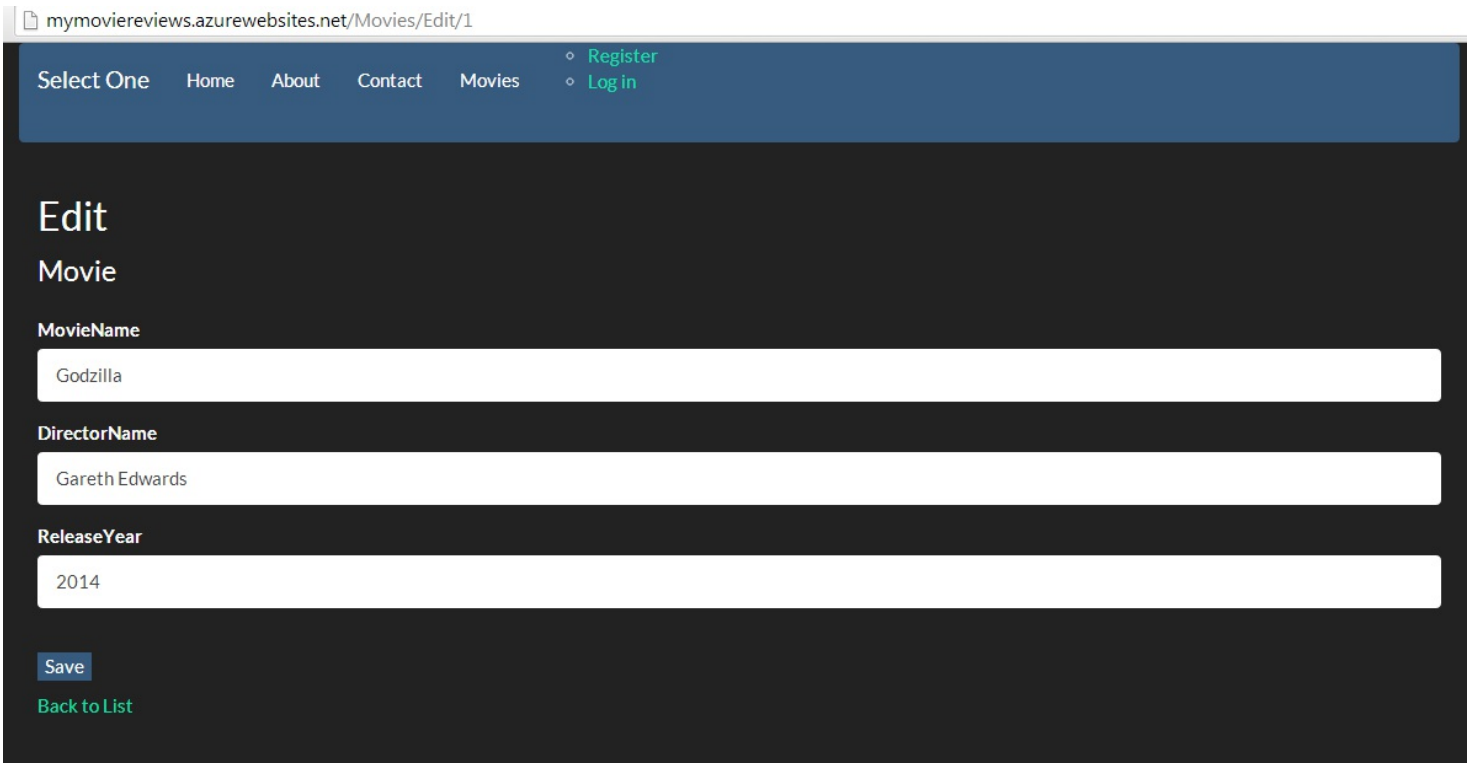
If you look at the URL, you will notice that I have already deployed the same on Azure (Microsoft's Cloud).



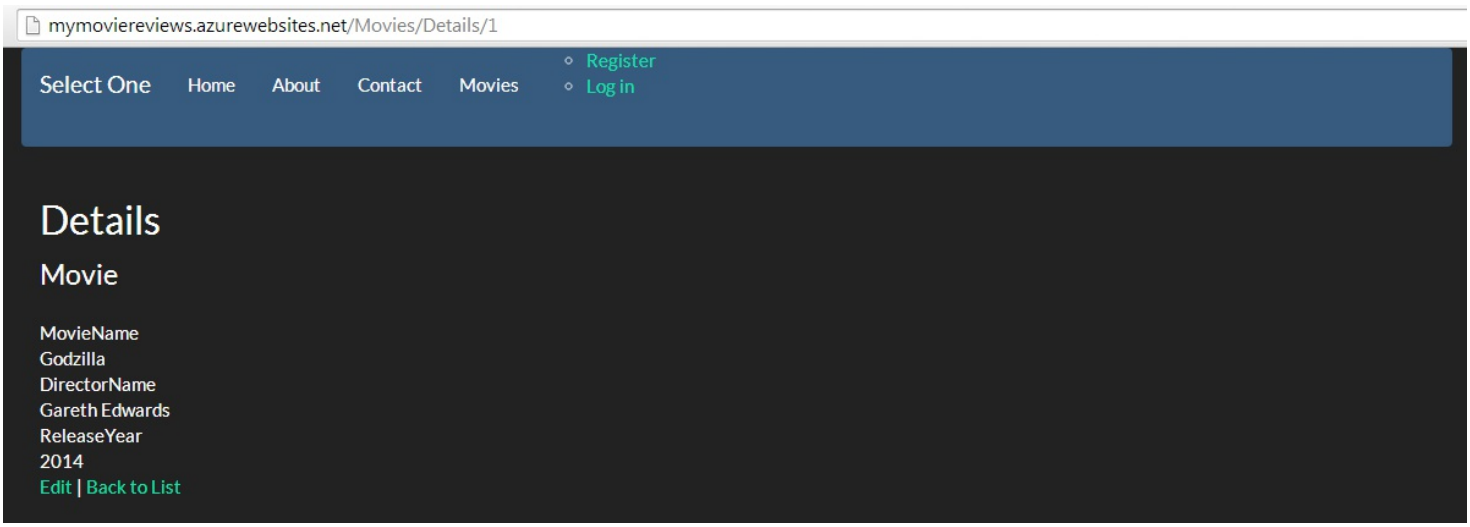
So, here we are at the home page listing couple of movies. It has got features like Paging, Async Search, Bootstrap Features and many more UI related stuffs which you will be exploring in the coming chapters.



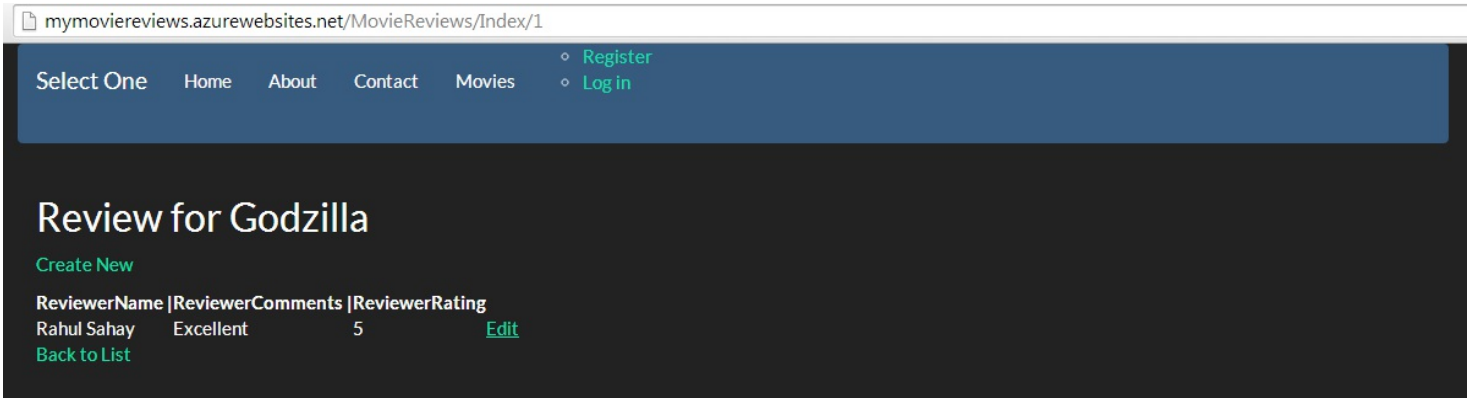
Now, the above page is quite descriptive. So, when you click the Edit link corresponding to the movie, it will emit text in editable mode as shown below.



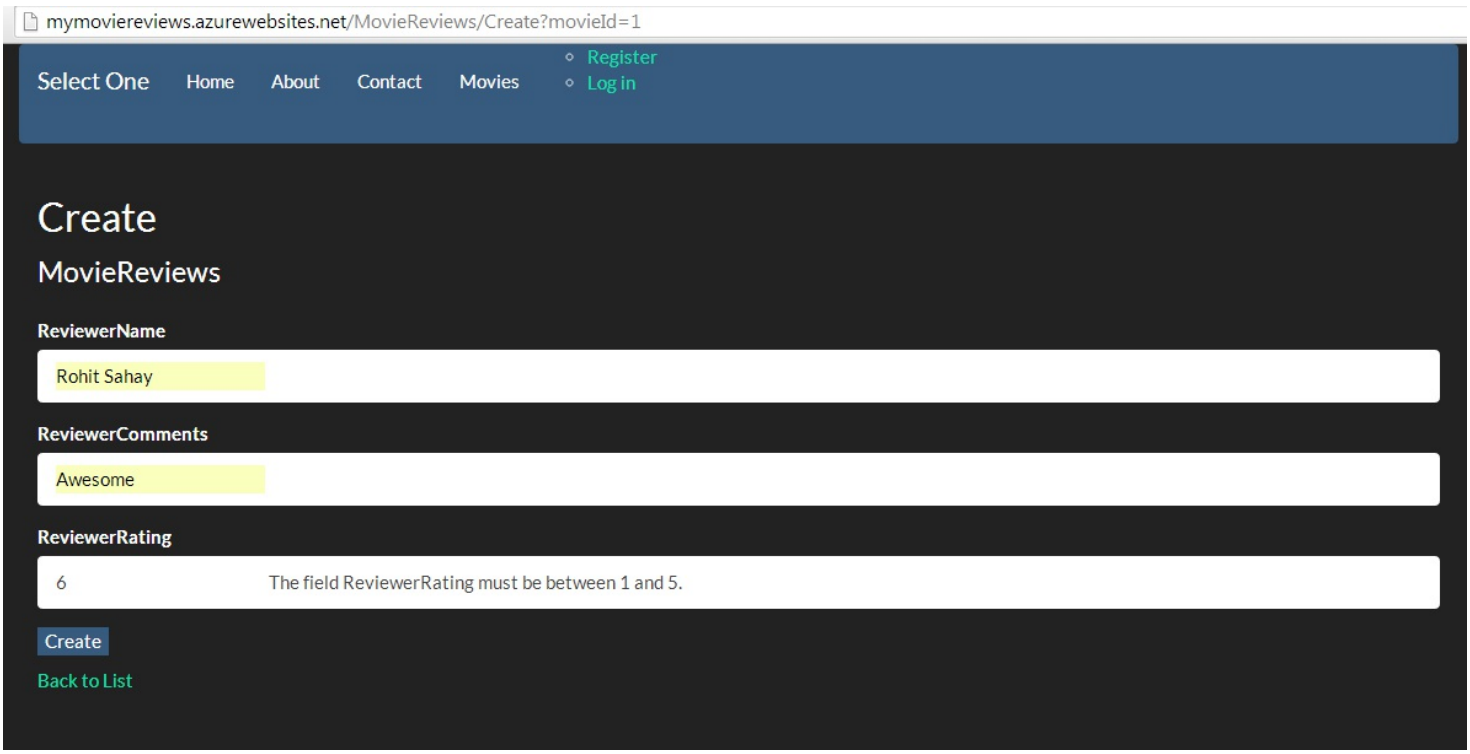
So, when you click on "***Back to List***", it will take you to the Movies List back and from there if you click the "***Details***" link, it will give the movie details as shown ahead.



Now, from here also you can edit the record. Now, back from Movies List if you click on "***Reviews***" link, it will take you the below page.

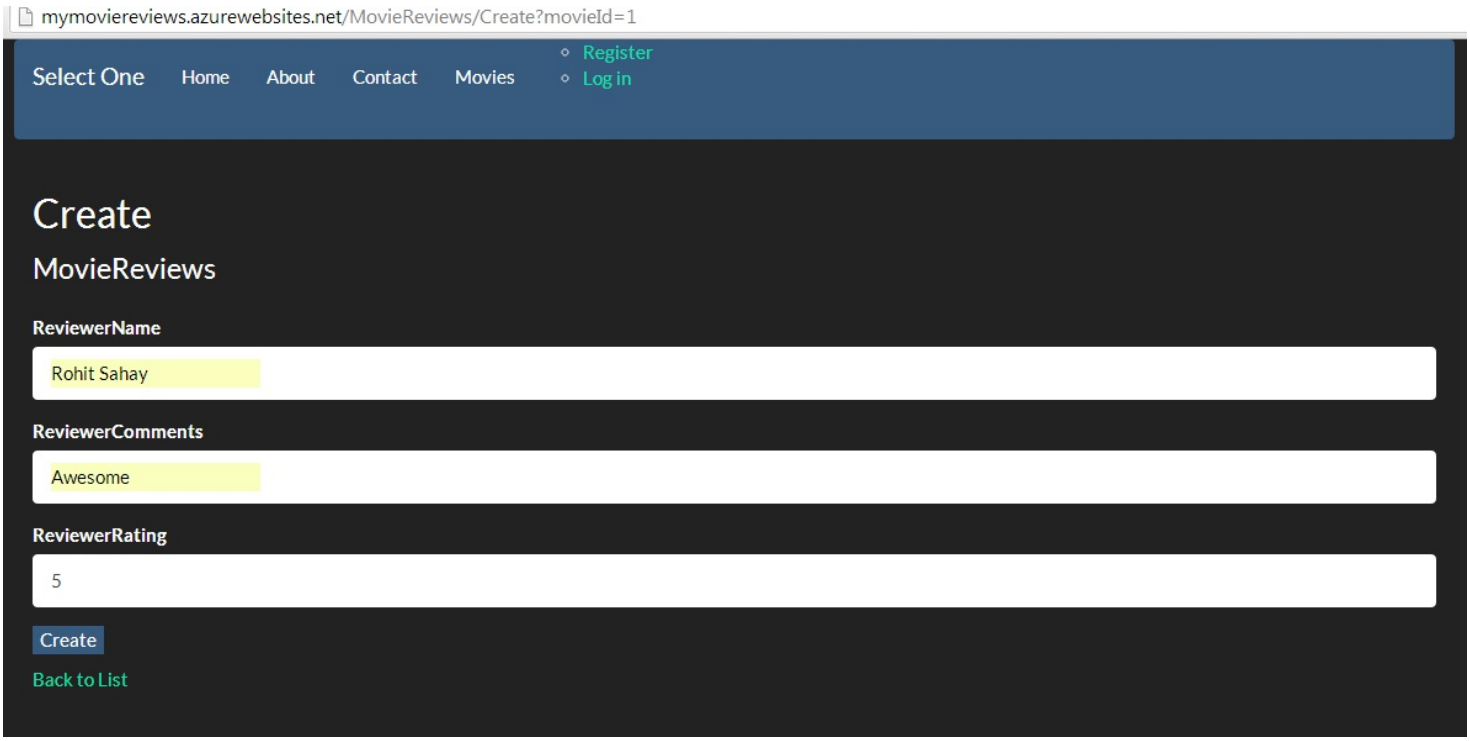


So, from here you can obviously create a new review as well as shown in the screen shot.



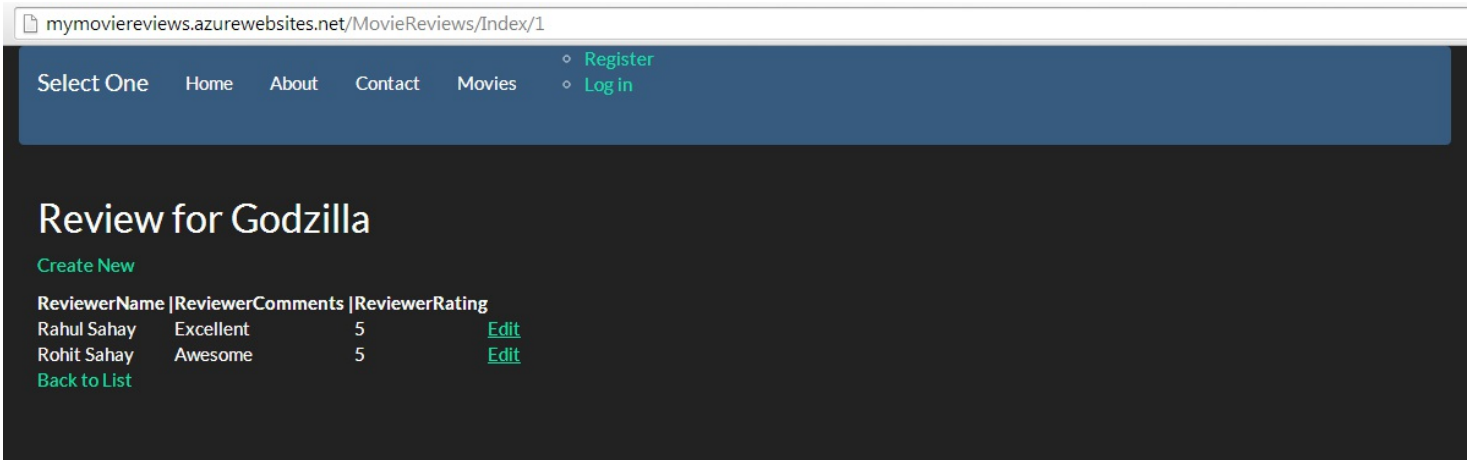
Now, as you can see in the above screen

shot as soon as user tried to enter a rating 6 which is greater than 5, it threw an error. So, at this point of time, you cannot submit this review. You need to fix this 1st and then submit as shown below.

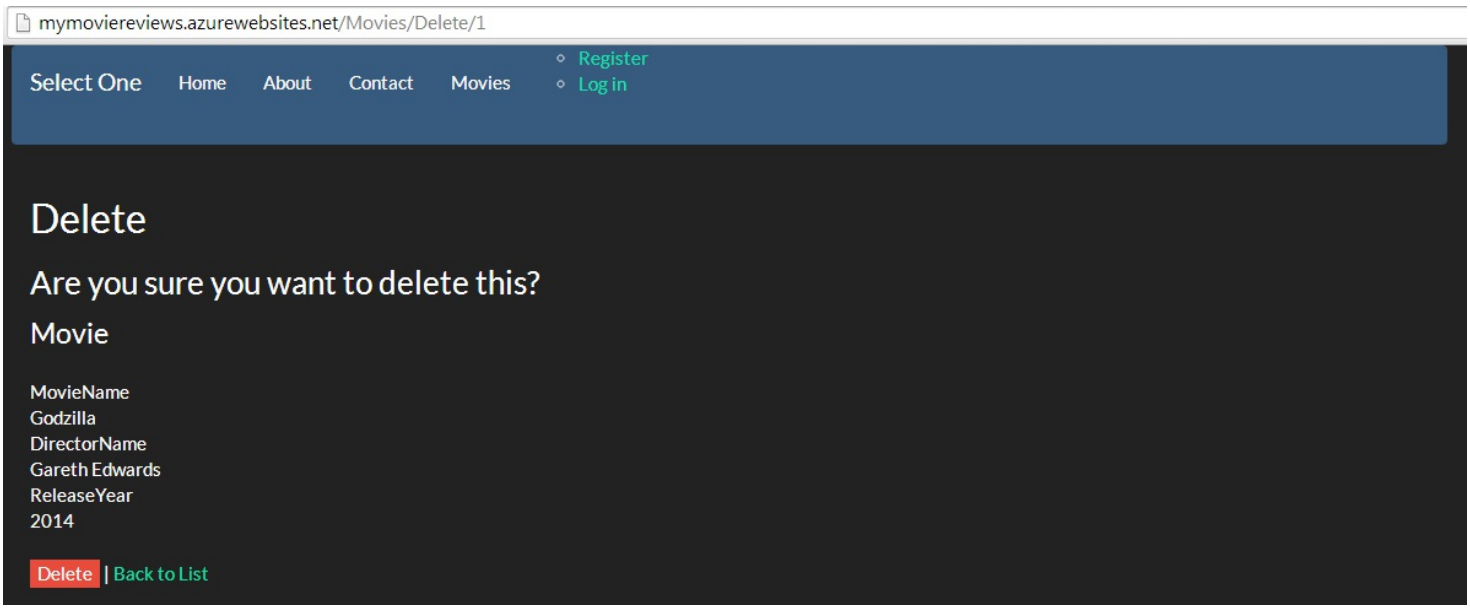


So, as you can see in the above screen shot, as soon as I corrected the error, error description wiped out

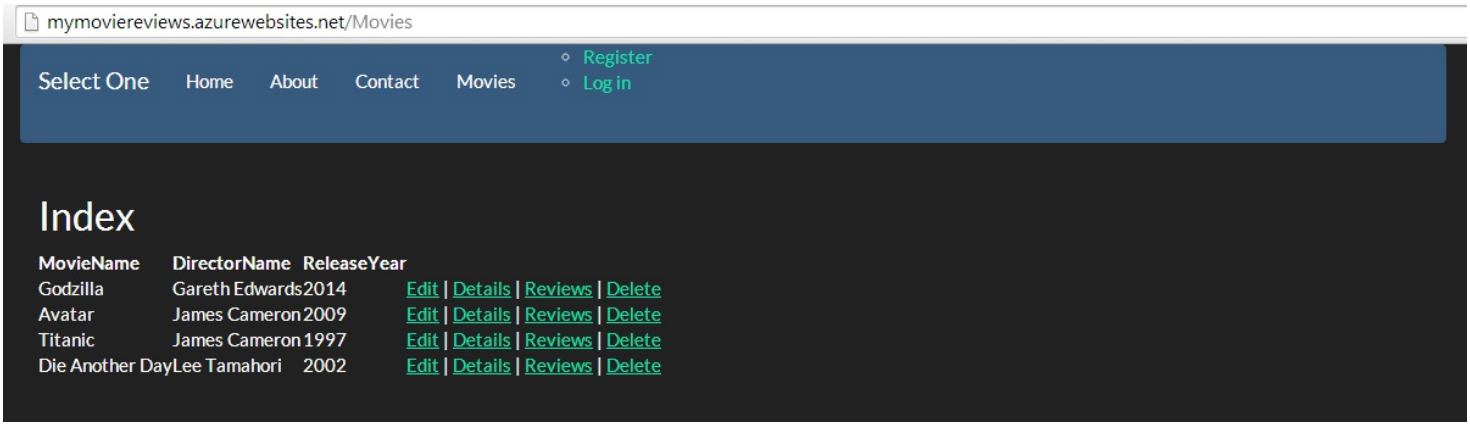
automatically. Now, I can go ahead and submit the record.



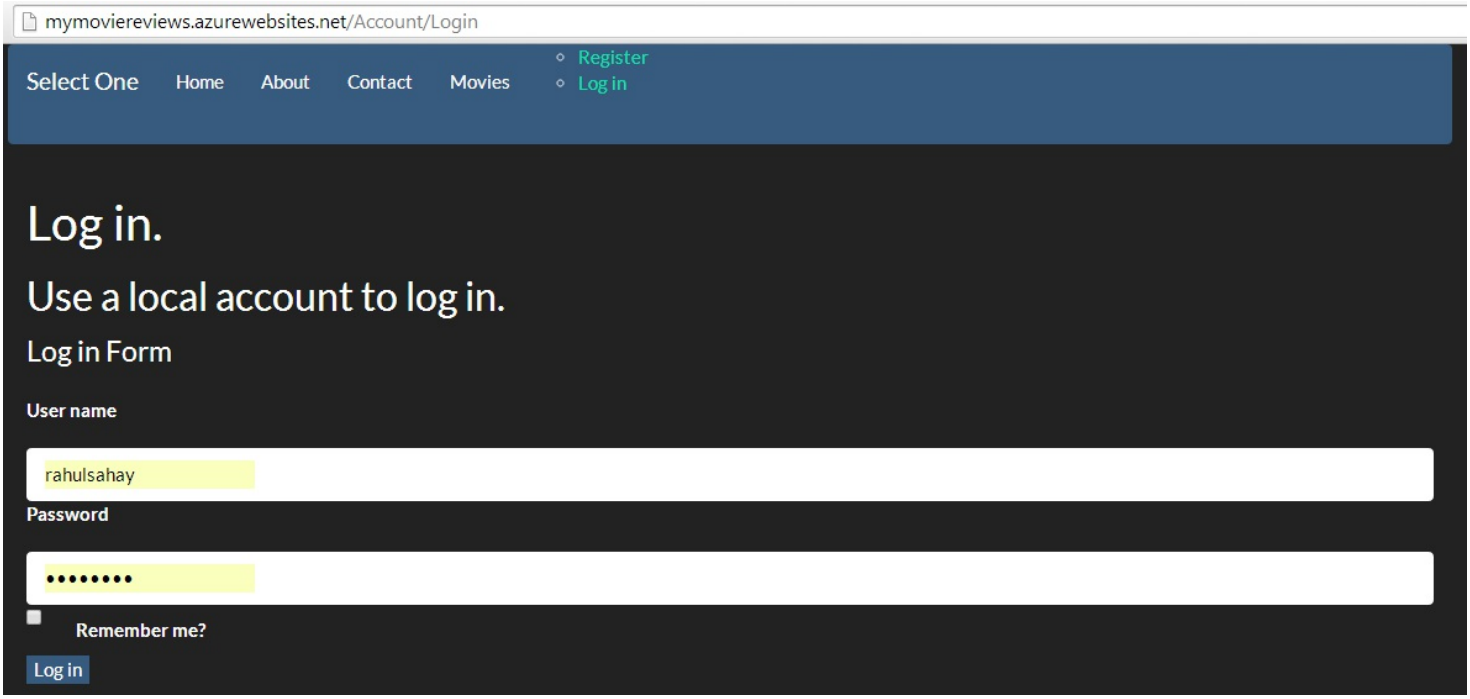
Now, if you click "***Delete***" option back from the list, then you will land up in the below shown delete page.



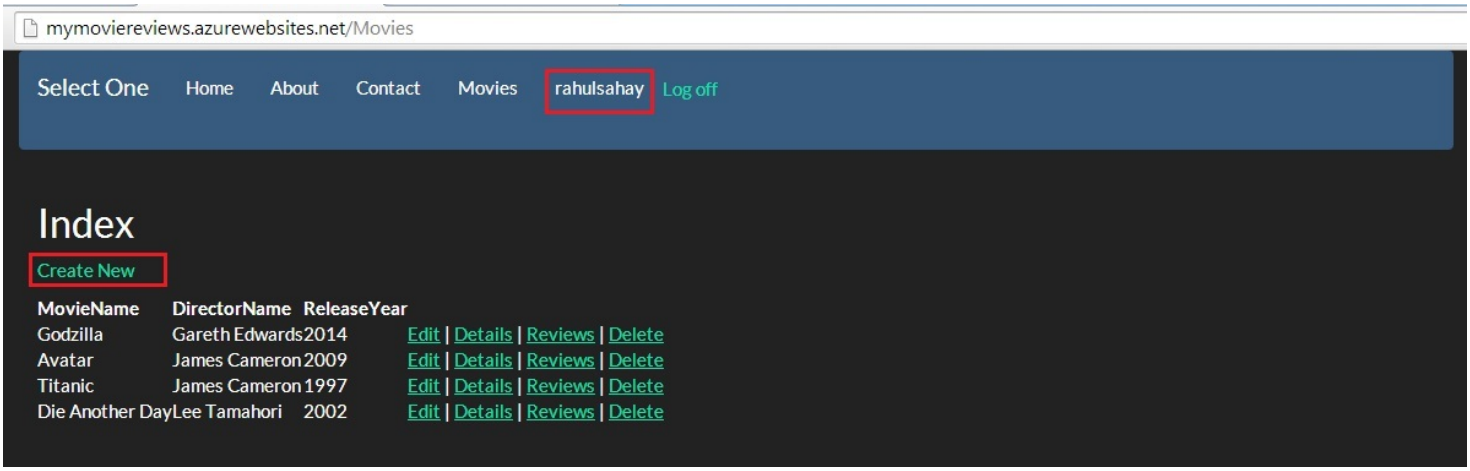
So, from here, you can go ahead and delete the page. Now, if you come back on the Movies List, you will notice that there is no option for creating the movie.



Well, for that I have made couple of changes like you should be authenticated 1st and authenticated as an admin, then only you will be able to see the Create Link. So, let me go ahead and login using my credentials.

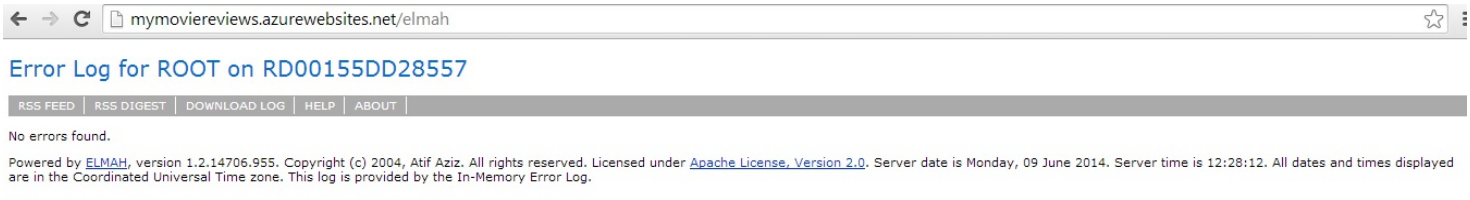


Then, it will show me the below Create Link.



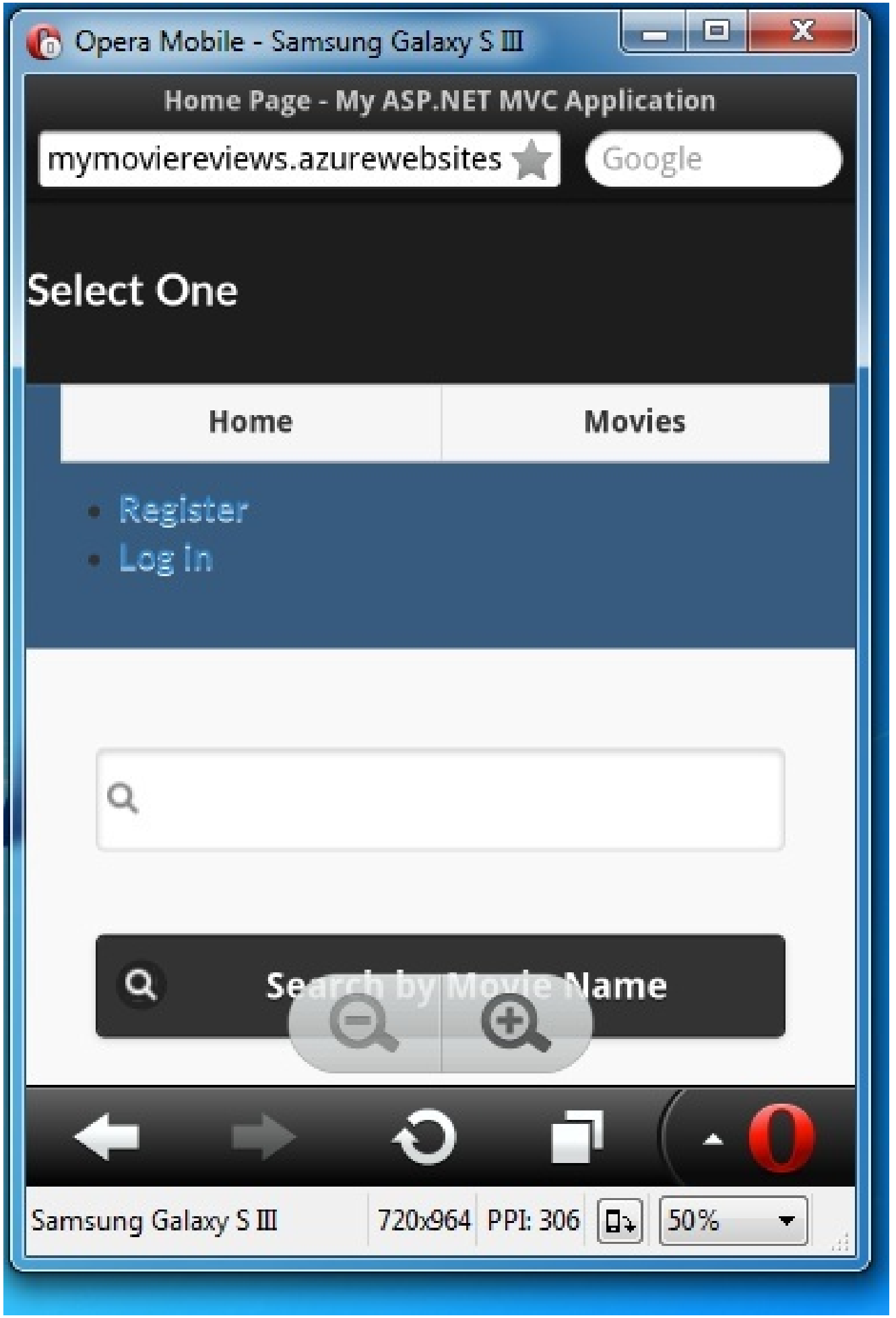
- So, with that Create link, only admin users can go ahead and

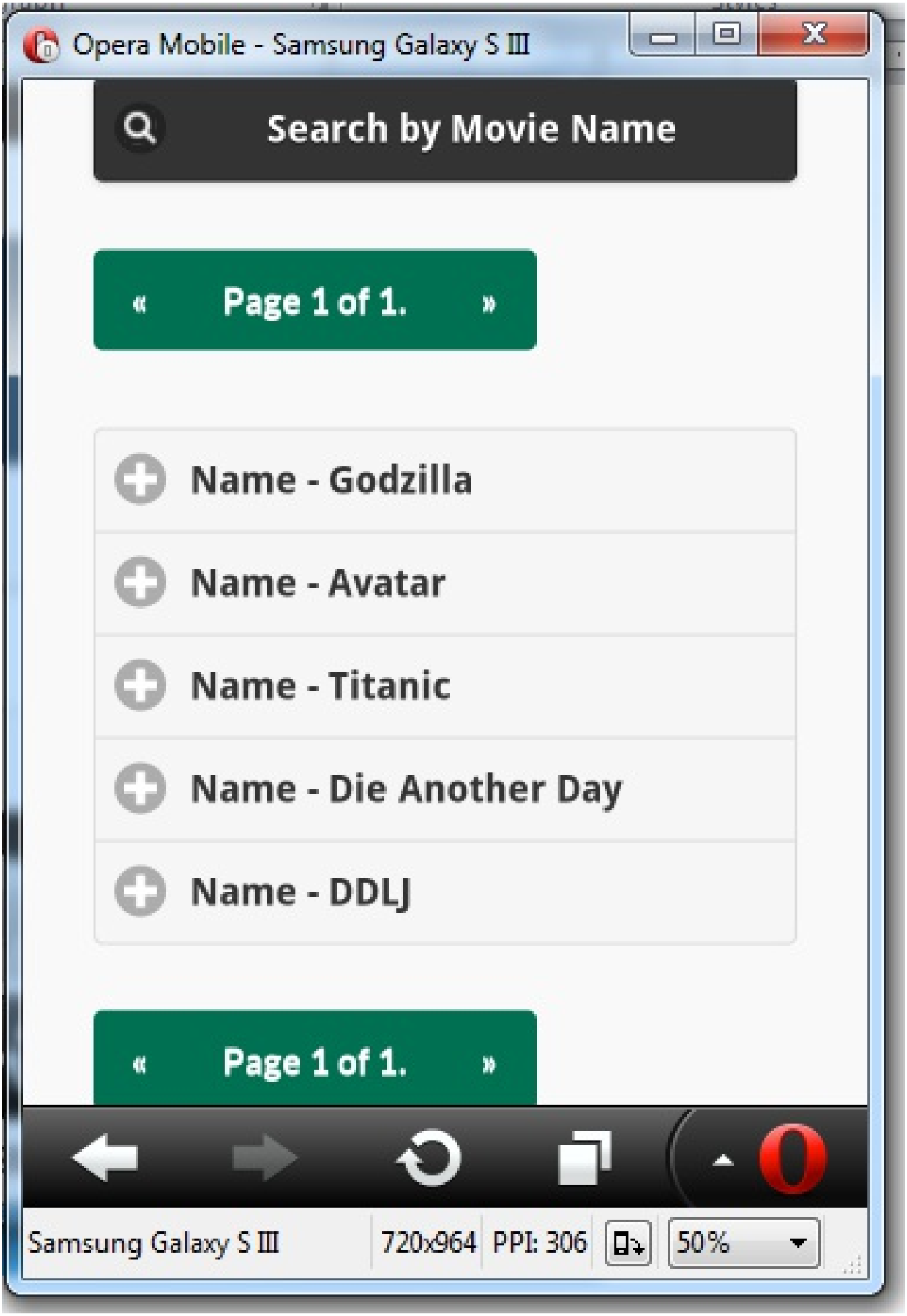
create the movies. So, this way I have restricted this action for anonymous users. Now, one more thing which I would like to show you is glimpse of my logging technique. I am a big fan of ***ELMAH Web UI*** which you learn in detail in coming chapters. But, for now just look at the glimpse of that. So, when you navigate *http://mymoviereviews.azurewe* then it will produce you the application snapshot as shown ahead.



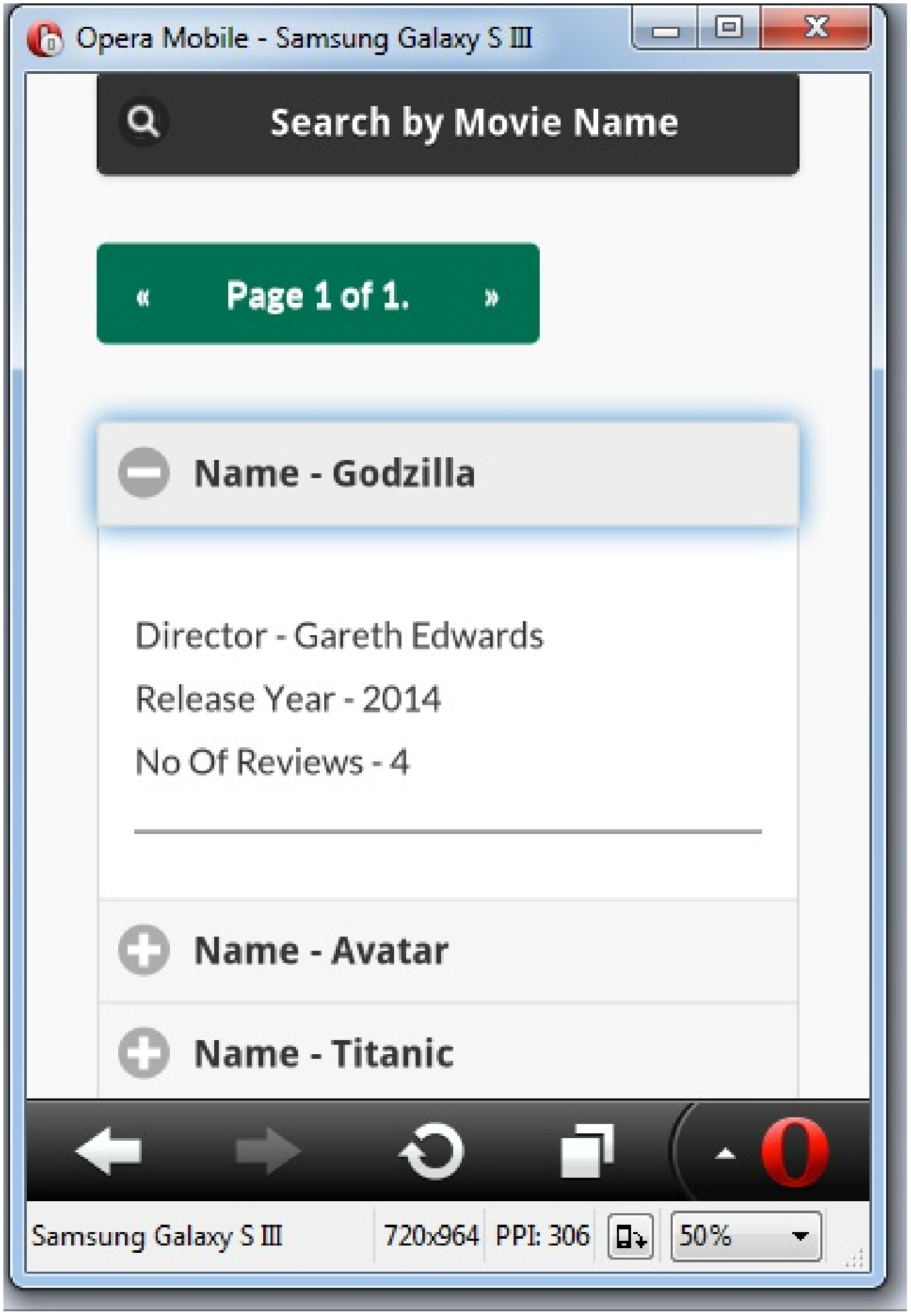
Currently, my application is running fine; hence it didn't produce any error log. Like this there are many more features which you can delve while building the application right from the scratch. So, stay tuned and get ready to build your 1st MVC application end to end.

Also, after creating the web application, I have converted the same app to support mobile browsers as well. So, now, if you open the same site in your mobile phone, it will get stacked as per your mobile resolution. So, the application has become smart enough to figure out that it is getting invoked from which device. Now, below are screen shots of this sample application.

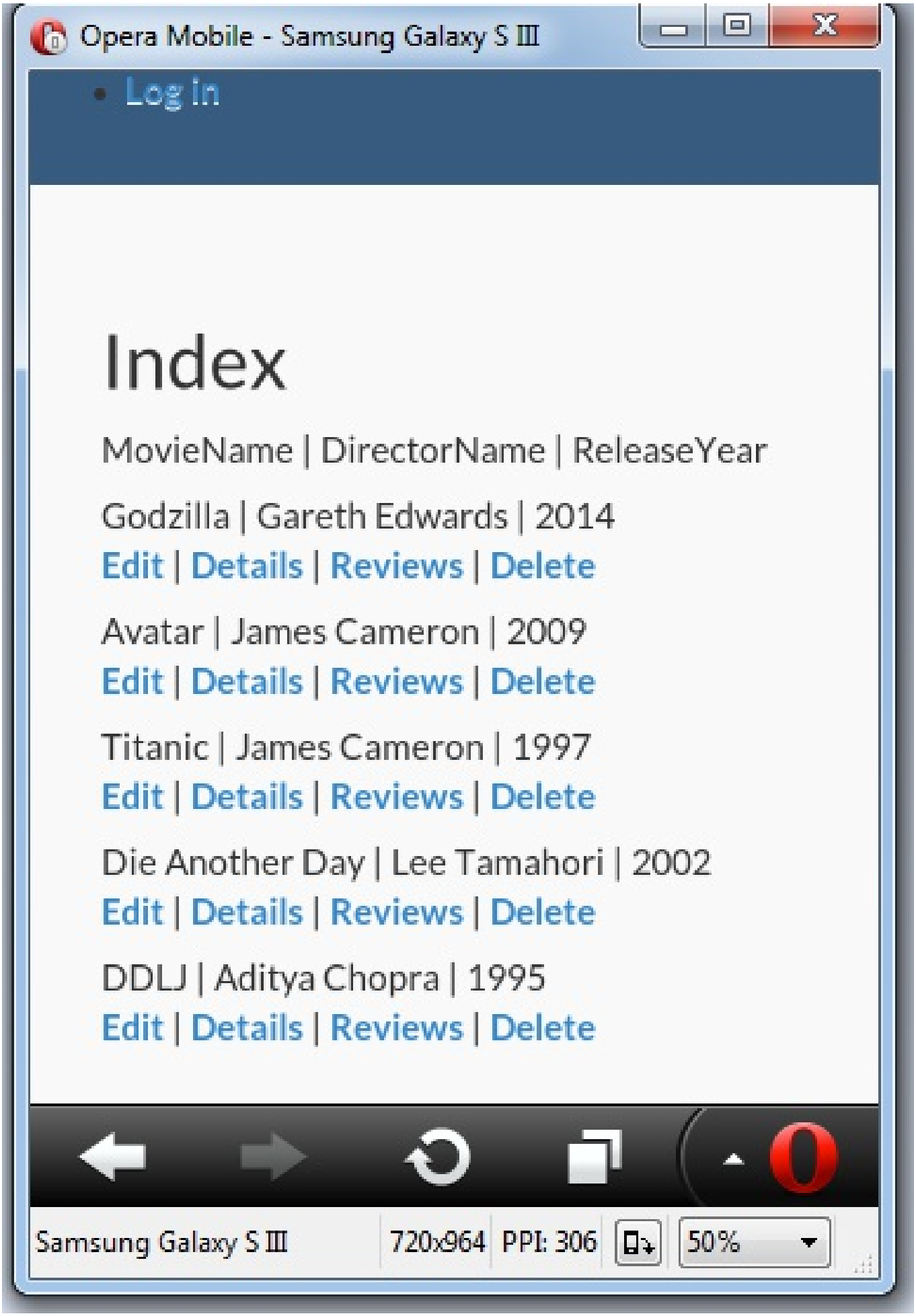

Opera Mobile - Samsung Galaxy S III
Home Page - My ASP.NET MVC Application
mymoviereviews.azurewebsites
Google
Select One
Home
Movies
Register
Log in
Search by Movie Name
Samsung Galaxy S III
720x964
PPI: 306
50%

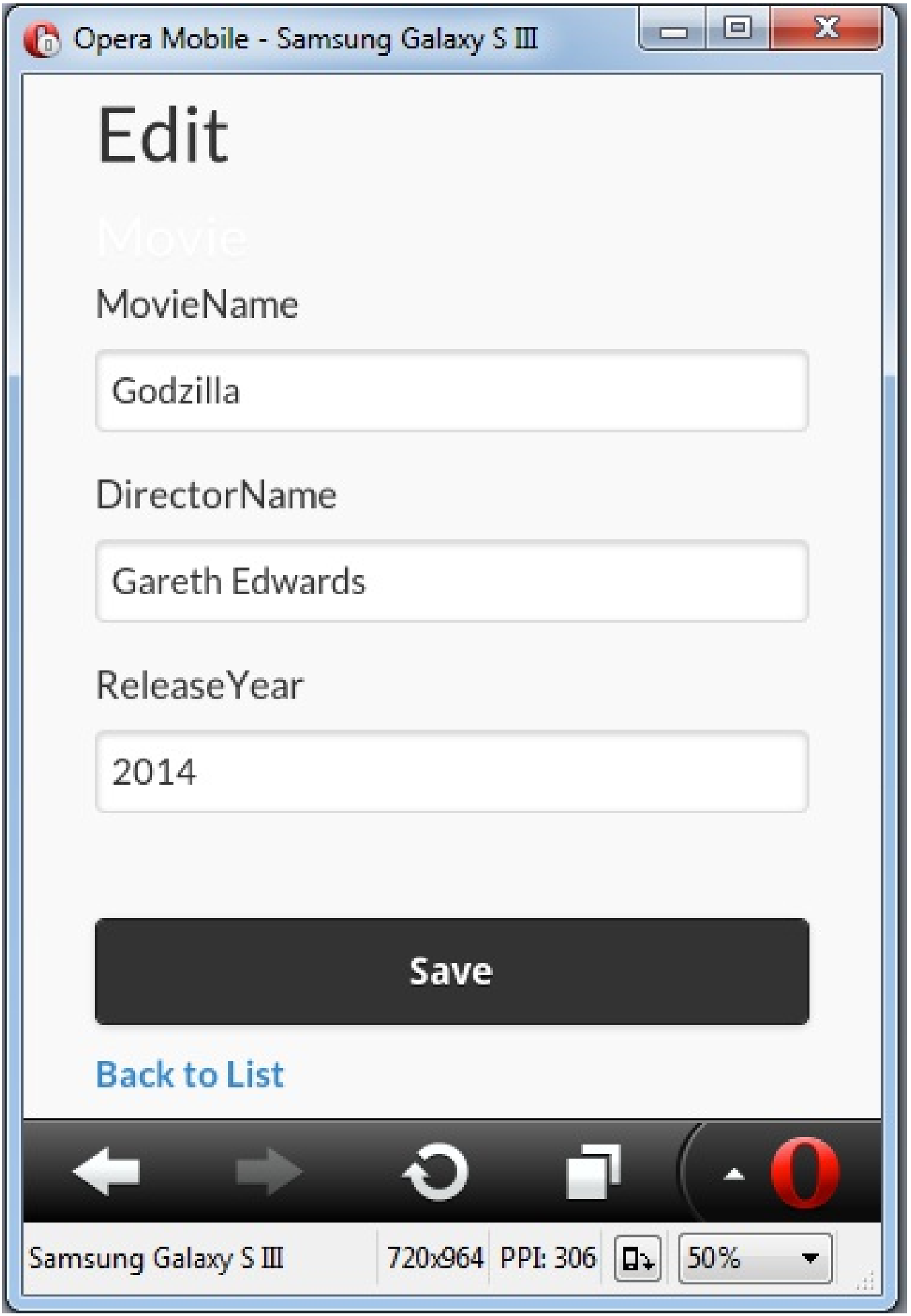

Opera Mobile - Samsung Galaxy S III
Search by Movie Name
« Page 1 of 1. »
Name - Godzilla
Name - Avatar
Name - Titanic
Name - Die Another Day
Name - DDLJ
« Page 1 of 1. »
Samsung Galaxy S III
720x964
PPI: 306
50%

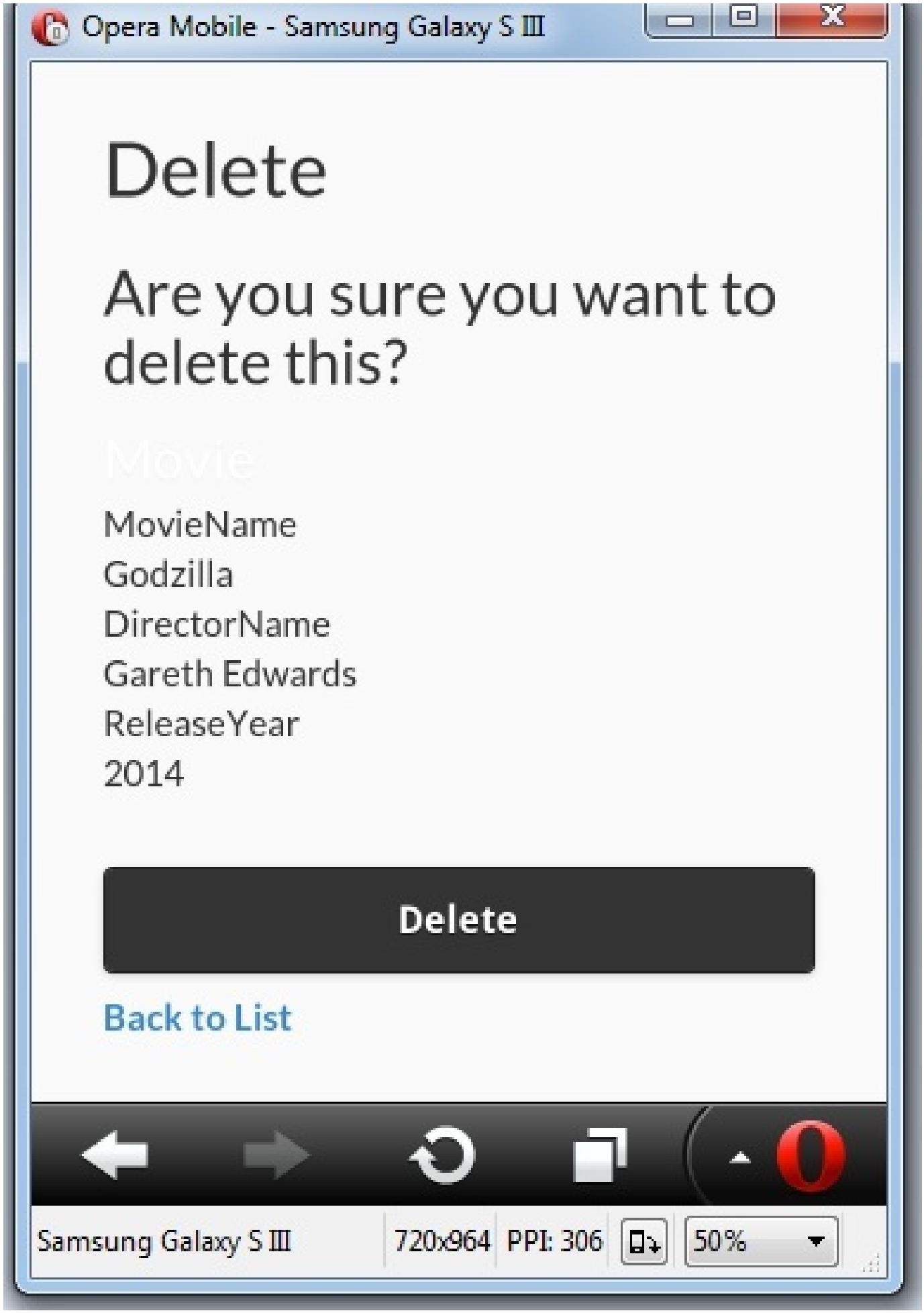
So, as you can see it has got different mobile user interface, and menu styling is also different. Also, the browser here I am using to test the same is opera mobile browser for desktop. Other features like collapsible panes as shown ahead fits very nicely with mobile devices.


Opera Mobile - Samsung Galaxy S III
Search by Movie Name
« Page 1 of 1. »
Name - Godzilla
Director - Gareth Edwards
Release Year - 2014
No Of Reviews - 4
Name - Avatar
Name - Titanic
Samsung Galaxy S III
720x964
PPI: 306
50%

On similar lines you can see other pages like movie and movie reviews as shown ahead.


Opera Mobile - Samsung Galaxy S III
Log in
Index
MovieName | DirectorName | ReleaseYear
Godzilla | Gareth Edwards | 2014
Edit | Details | Reviews | Delete
Avatar | James Cameron | 2009
Edit | Details | Reviews | Delete
Titanic | James Cameron | 1997
Edit | Details | Reviews | Delete
Die Another Day | Lee Tamahori | 2002
Edit | Details | Reviews | Delete
DDLJ | Aditya Chopra | 1995
Edit | Details | Reviews | Delete
Samsung Galaxy S III
720x964
PPI: 306
50%


Opera Mobile - Samsung Galaxy S III
Edit
Movie
MovieName
Godzilla
DirectorName
Gareth Edwards
ReleaseYear
2014
Save
Back to List
Samsung Galaxy S III
720x964
PPI: 306
50%

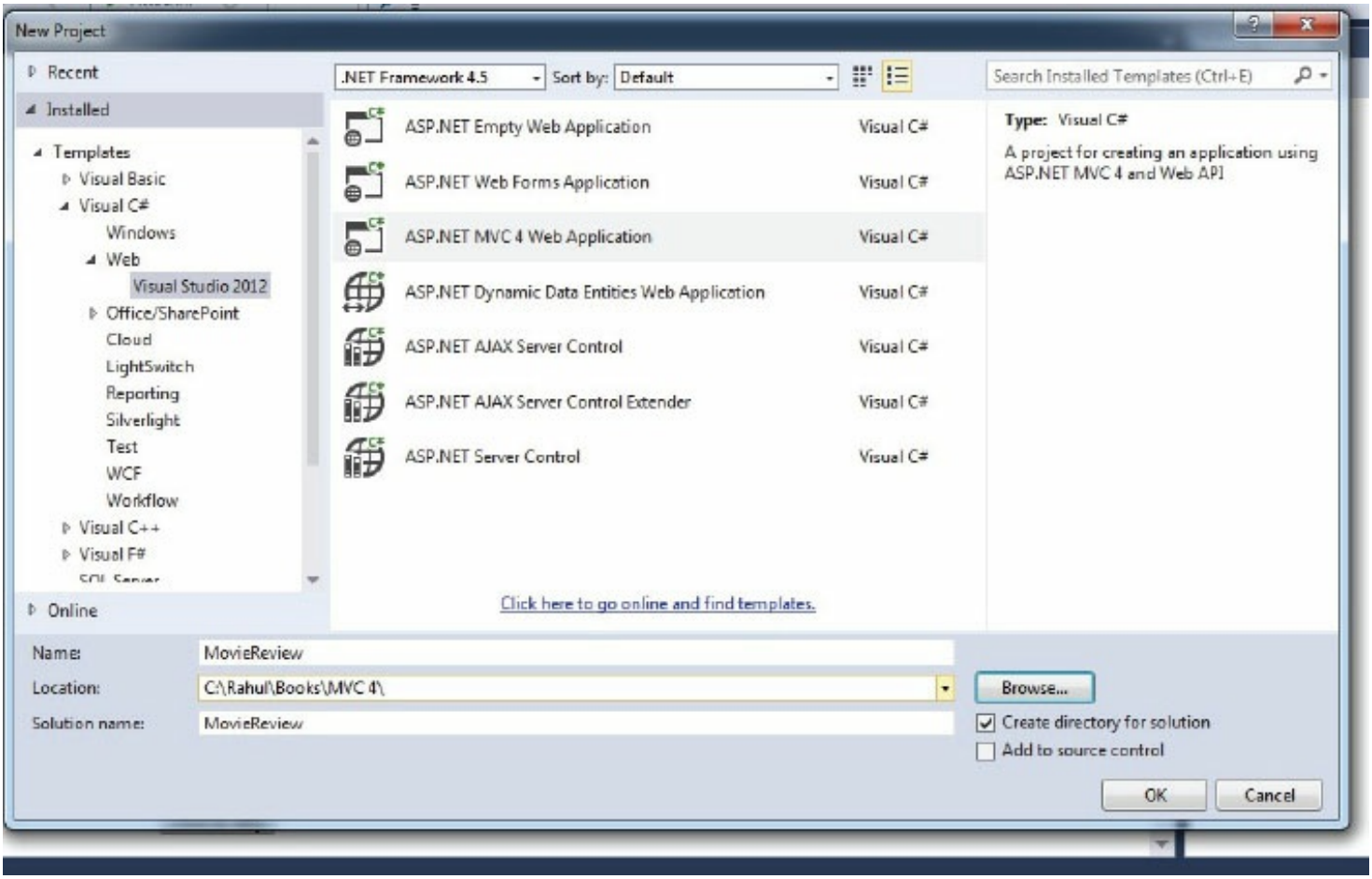

Opera Mobile - Samsung Galaxy S III
Delete
Are you sure you want to delete this?
Movie
MovieName
Godzilla
DirectorName
Gareth Edwards
ReleaseYear
2014
Delete
Back to List
Samsung Galaxy S III
720x964
PPI: 306
50%

So, this was the application glimpse in a nutshell. So, during this entire course you will be learning how to build an End to End Application which will not only support desktop version browsers rather mobile as well.

Creating a MVC Application:-

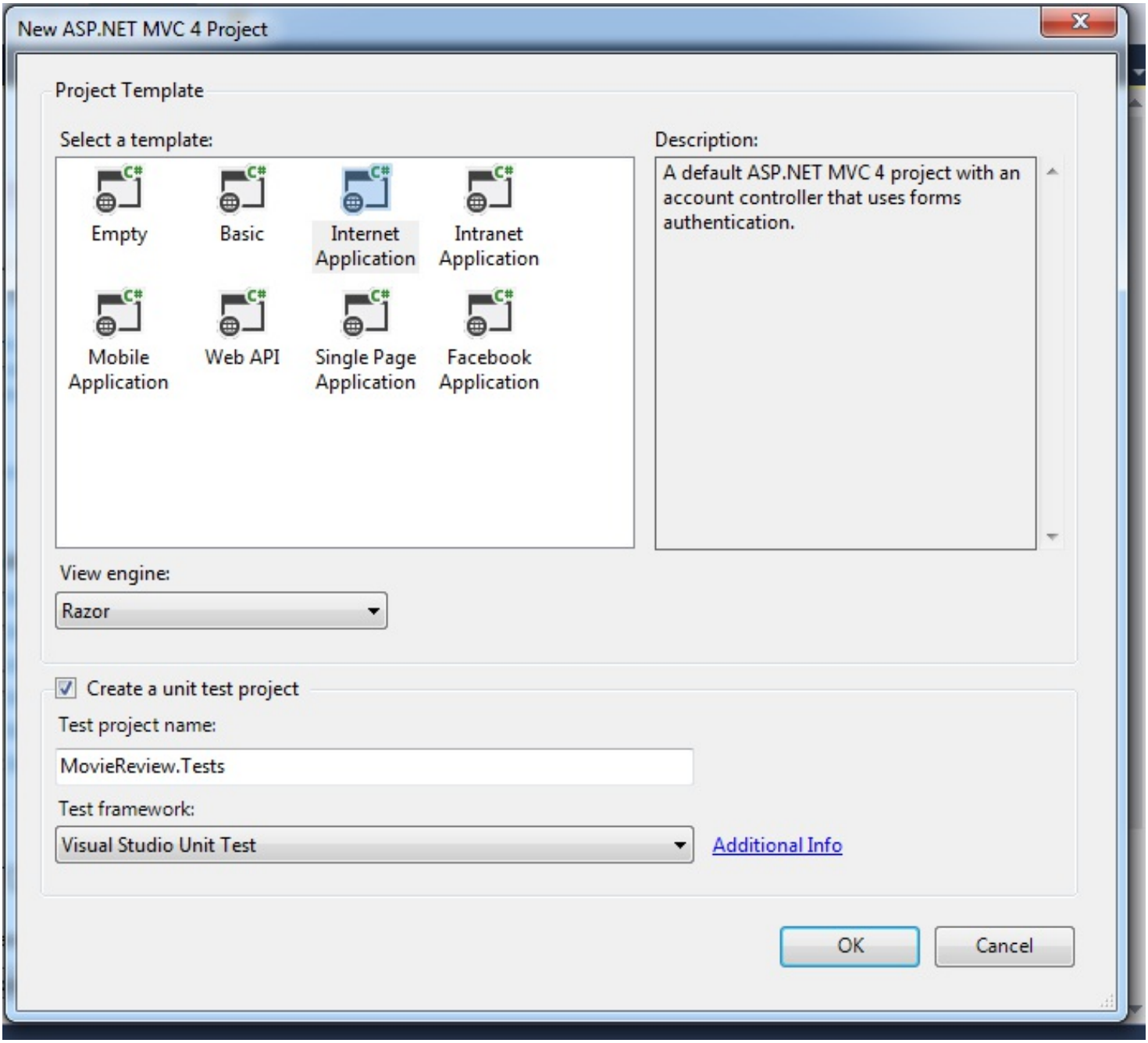
I always believe to learn anything is to get started with the hands-on the same, so that way you will understand the practical approach of solving any problem statement. However, we'll be building a simple web app for "***Movie***

Review" where in I'll walk you through all the tiny steps involved in building the site. So, let's get started. Before we begin, we must install the prerequisites for making this app. Now, to create a MVC Application, open the Visual Studio, click on New Project and select the below mentioned project as shown in the snapshot.



Note: - Here am using Visual Studio 2013 Ultimate Version.

Now, as soon as you select the MVC 4 web application, it will present couple of different application templates option as shown ahead in the screen shot.



Empty Template:- Empty template will just have the basic folder structure that's it and nothing more than that.

Basic Template:- The basic template

will give you the MVC infrastructure in the solution. However, basic templates are for experienced MVC guys who want to customize the solution their way.

Internet Application:- Internet Application is the one which I am interested in as you can see in the above screen shot. However, it will give all the required dependencies in my solution structure to get started with my web application.

Intranet Application:- Intranet application is the one which deals with Windows Authentication. So, once you select this option, your app gets automatically configured with windows

related settings.

Mobile Application:- Mobile application is the one which is included with MVC 4. The mobile application is preconfigured with JQuery Mobile. It helps developer to create just mobile sites. It includes themes which is supported by mobiles, touch enabled UI etc.

Web API: - Web API terminology will be discussed in detail later in this book. However, for now you just need to know that Web API is basically a framework which supports creating HTTP services.

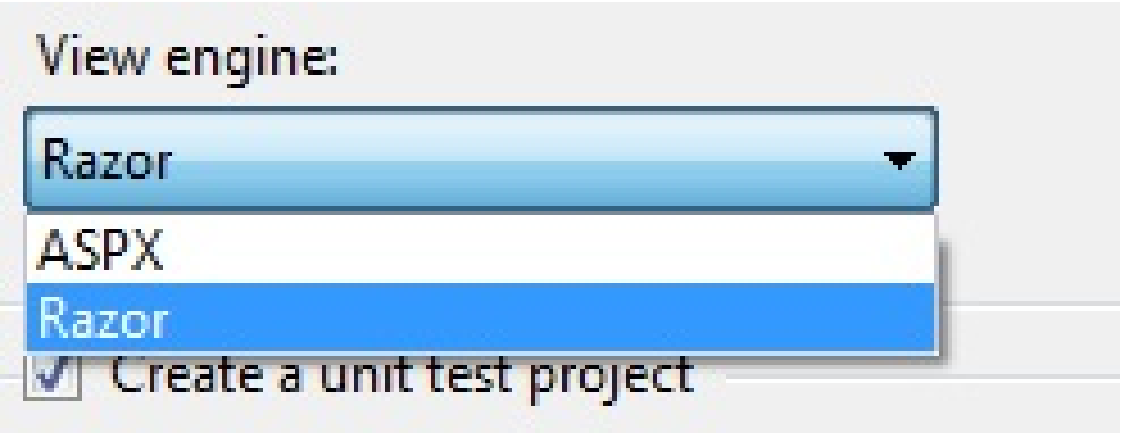
Single Page Application: - SPA is a

new terminology which offers building Single Page Application, focused mainly on client side interactions using tons of JavaScript and different JS Frameworks like Knockout, Durandal, Angular etc. This kind of web application is highly interactive and feature rich for ex: - Gmail, Outlook etc.

Facebook Application: - Facebook application is I would say a kind of API support which offers developer to use the Facebook API in their application to build a Facebook centric website. This is really a cool stuff.

Now, the next option which I have selected above is razor as view engine,

so if I click on this dropdown I could see two options available as Razor and ASPX.



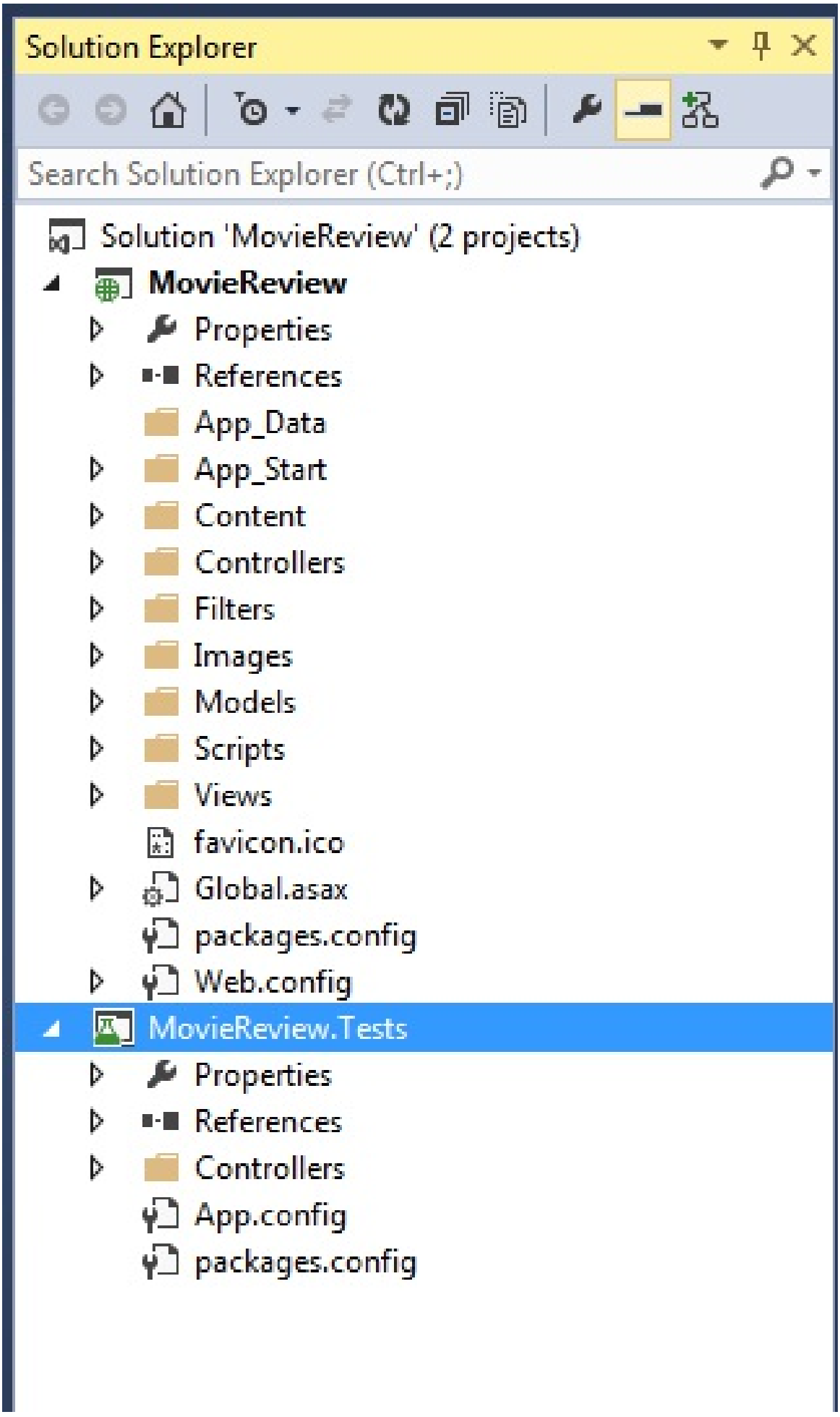
I also have a habit of checking Unit Test Project as you can see below in the screen shot. So, upon checking this checkbox, Visual Studio will create one more project for me in the solution which is going to be Test project. Here, I can go ahead and write my test cases against any specific module or scenarios.



So, when you click on the dropdown of Test framework, you will see only one test framework in there. However, Microsoft has given an option to developers to install their friendly testing framework like NUnit, XUnit, etc. So, once these frameworks will get installed, this will get added in the dropdown box.

How MVC Applications are Structured:-

Below is the snapshot of the MVC application structure. This gives a brief idea that how your simple MVC app is structured initially.


Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'MovieReview' (2 projects)
MovieReview
Properties
References
App_Data
App_Start
Content
Controllers
Filters
Images
Models
Scripts
Views
favicon.ico
Global.asax
packages.config
Web.config
MovieReview.Tests
Properties
References
Controllers
App.config
packages.config

Below I have listed specific directories which are there in the solution and their meaning.

Models:- Section containing all the classes related for fetching the data and manipulating the same.

Views:- Section containing all the UI related stuffs. These are the ones which are going to be rendered based on controller's action.

Controllers:- Section where in the entire controller classes are placed which basically handles all the incoming requests from the browser.

Scripts:- Section where in all the scripts related stuffs are placed.

Images:- Section where in all the images are placed which is used across your site.

Filters:- Section where in all the filters are placed. We'll do deep dive usage of filters later in coming chapters.

App Data: - Section where in all the data files are stored meant for reading and writing the same.

App Start: - Section where in all configurable stuffs resides like Web API, Routing etc.

Note: - This is basic internet app MVC structure. However, please note that MVC Framework never enforces any kind of folders organization. You can obviously customize the folders your way, like in many enterprise projects Models go in a separate class library project just to make sure reusability. However, you could do higher level of separation, it entirely depend on you.

SUMMARY:-

So, in this chapter we have just seen the overview of MVC, history of MVC and its different releases. We also understood that ASP.NET MVC is

another technique of building web application, a completely different strategy rather stateless technique to build the web app. We have also seen the application glimpse which we are going to build right from the scratch and host the same on cloud. As we progresses module by module in different chapters your doubts regarding MVC development will be getting clearer.

Chapter 2: Understanding Controllers

WHAT DO you find in this CHAPTER?

⇒ Introduction

⇒ Controllers

⇒ Working with your 1st controller.

⇒ Action Results

⇒ Action Selectors

⇒ Action Filters

⇒ Summary

Introduction:-

Hi friends, I hope you have enjoyed the application glimpse in the last section and you are ready to jump start to create

your 1st MVC application from blank slate. So, basically in this section we will understand the concepts of controllers like why it is there in MVC, what its job and why it is often referred as central unit of your application.

Controllers:-

Controllers are basically nerve of ASP.Net MVC as it is going to be the 1st recipient which is going to interact with incoming HTTP Request. So, controllers are going to decide which model is going to be selected, then taking the data from the model and passing the same to the respective view, hence finally view

is going to be rendered. So, controllers in a nutshell are basically controlling the overall flow of the application taking the input and rendering the proper output.

Since, we have selected internet template while building the application, so we have been presented with two default controllers called as

Home Controller: - It will render the home page of the application.

Account Controller: - It will render the Login/Registration page of the application.

Below, is the sample snapshot of the Home controller being created by the

Visual Studio.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace MovieReview.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            ViewBag.Message = "Modify this template to jump-start your ASP.NET MVC
application.";

            return View();
        }

        public ActionResult About()
        {
            ViewBag.Message = "Your app description page.";

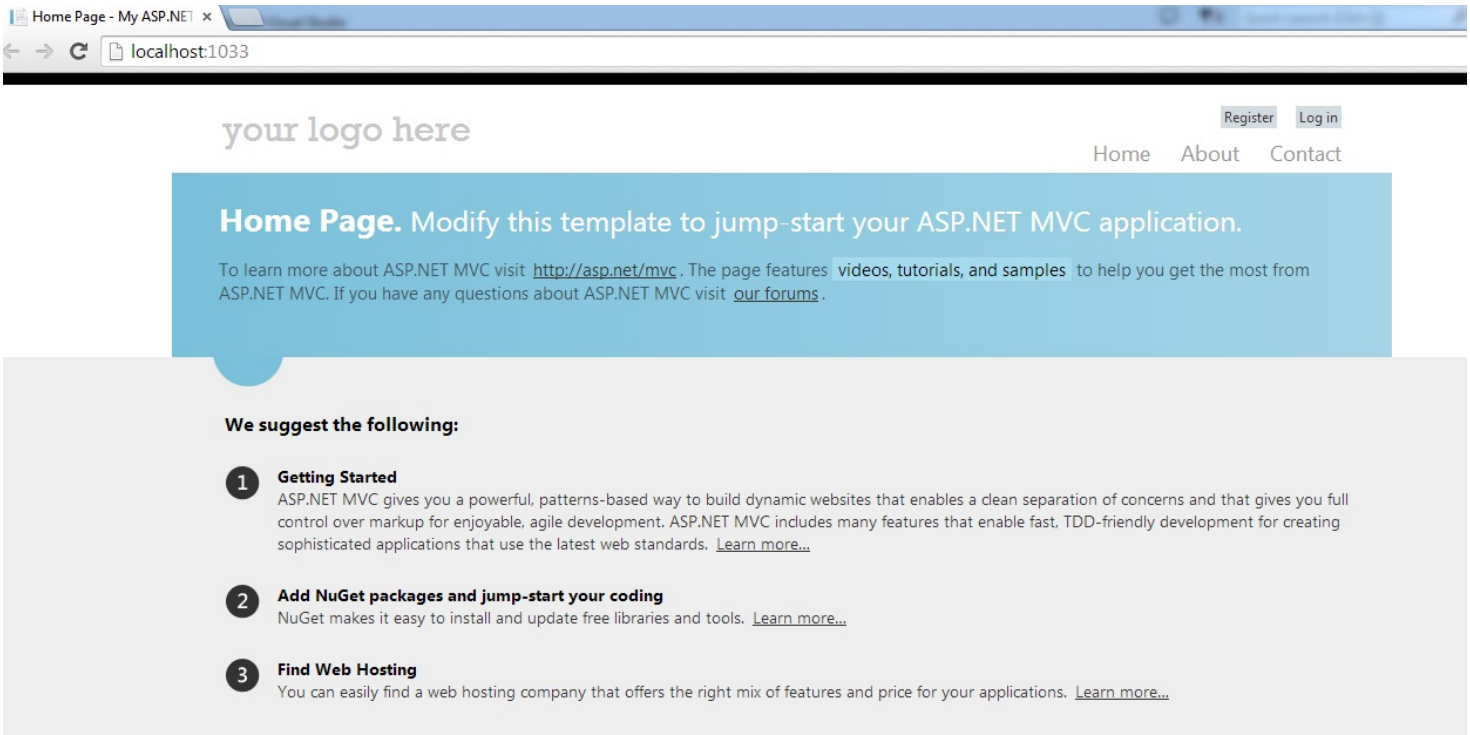
            return View();
```

```
        }

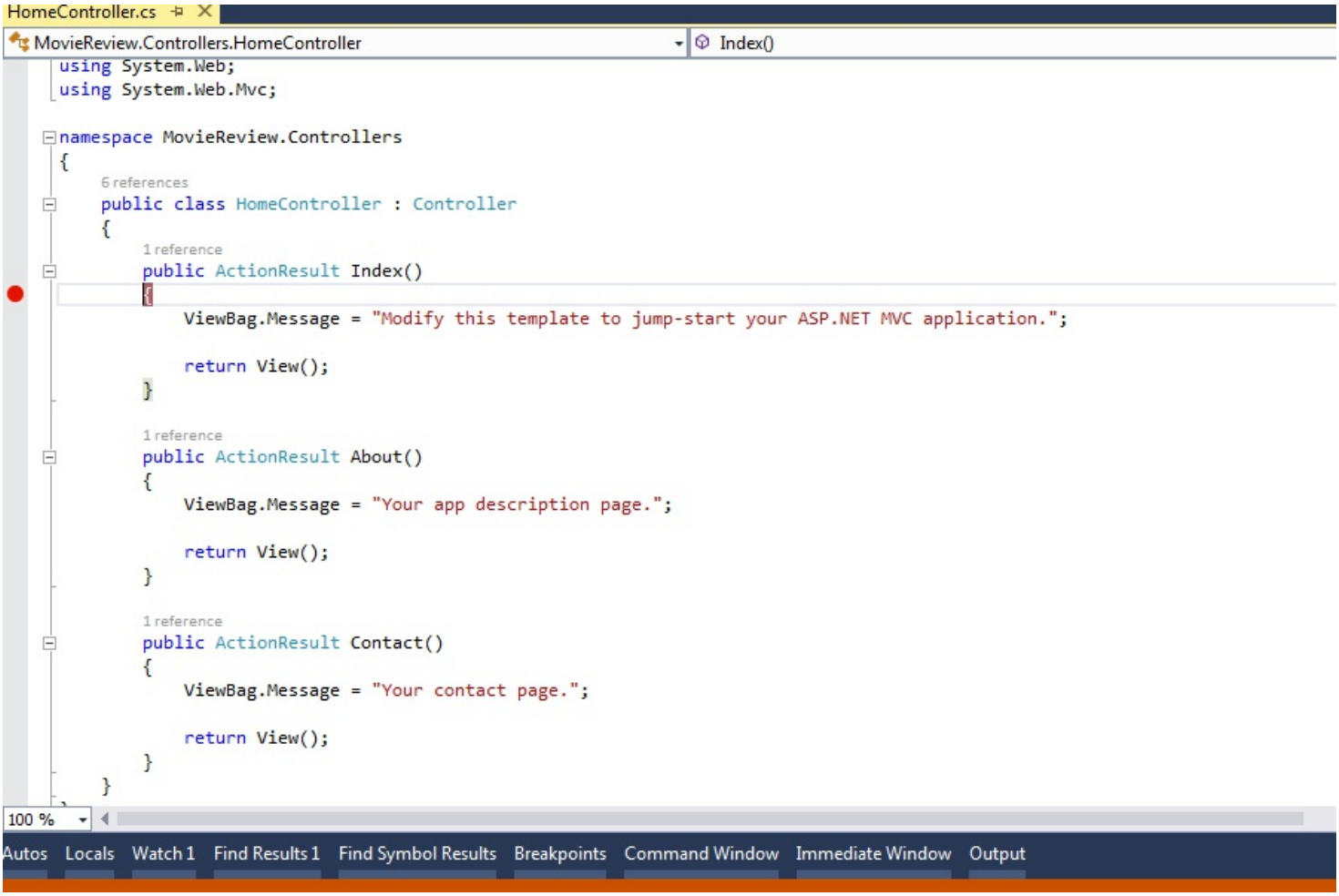
        public ActionResult Contact()
        {
            ViewBag.Message = "Your contact page.";

            return View();
        }
    }
}
```

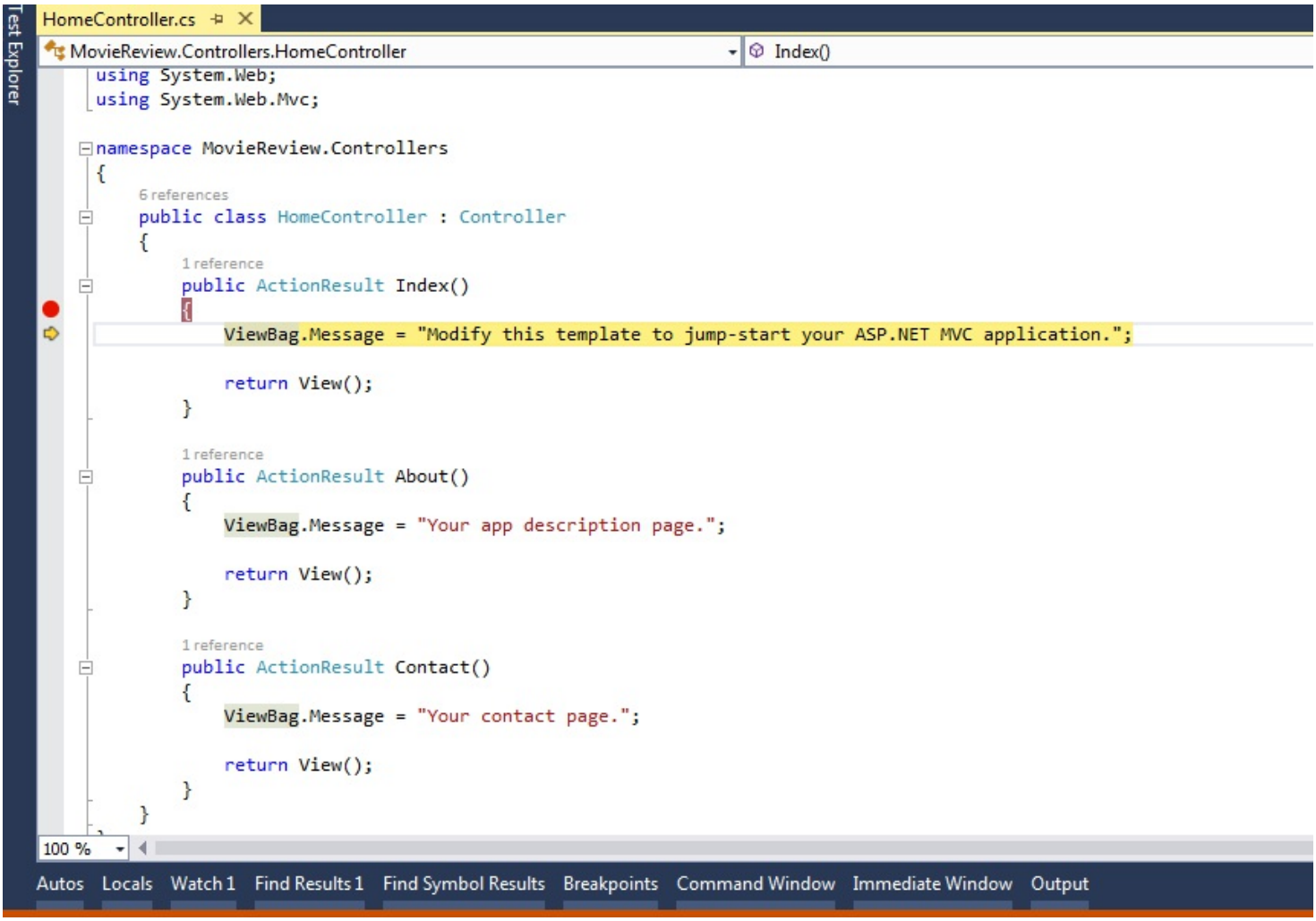
Now, when you run the application by clicking F5 button, it will simply render the Home view with the above welcome message printed on screen as shown below in the snapshot.



However, you can verify the same by putting one debug point in the home controller and see how the control is flowing and how view is getting rendered over here.

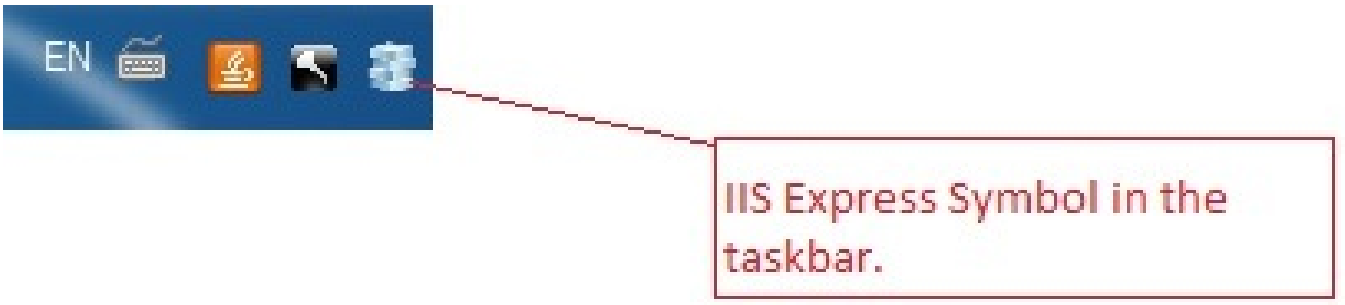


Now, when you run the application, you will see that control has hit the break point and then it rendered the corresponding view with the above details and the static details which is mentioned in the view.



We'll discuss View in more detail in the next chapter. But for now just understand that Home Controller's Index action is going to return the Index View under the Views ' Home directory. There is one more important point to notice here, that development box is running on the IIS

Express.



Now, when you right-click on IIS Express you will see the currently running application on it and the random port which is allotted to it.



IIS Express is basically development

version of Visual Studio which comes with Visual Studio 2012+ platform. It allocates random port for your website. In our case this is http://localhost:1033/.

Working with your 1st Controller:-

Let's get started by creating your 1st controller. This controller will simply search movies. Then, subsequently we'll start adding some more features to the item displayed on the screen like when you click on the movie; corresponding movie details will flash on screen. However, it's worth time spending to understand the routing before jumping to

write our 1st controller. Obviously, we'll do deep dive about routing in routing chapter. But, 1st let's understand few basic concepts of routing.

One of the core questions which you must be thinking that how ASP.Net MVC knows to load the index action when the URL http://localhost:1033/ is invoked, and the answer lies in the routing engine. The **Routing Engine** is the core concept of the ASP.Net MVC; it's not coupled with MVC Framework. Why am saying this, because we can use the routing engine to route the request for Web Forms, WCF Services etc. But, here in MVC we use this routing engine to redirect request to our controllers. In

order to execute the same, we give the routing engine a map to follow using "MapRoute" API.

```
routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Home", action = "Index", id =
UrlParameter.Optional }
            );
```

So, basically what a route map does, it provides a friendly name to the route, a pattern to follow and default parameters for the route. So, if you think about the job of the routing engine, its job is pretty simple. Its job is to examine the URL and figure out where to send the request for processing. When, it examines the URL, it picks up the little pieces from the URL which tells the routing engine, where to send the request. In an MVC

Application framework, we provide nomenclature "***Controller/Action/Id(somedata)***". So, if the routing engine doesn't find the specific piece of data inside the URL, like controller name or action name, then it can use the default values assigned in the Map. In the above piece of code, it is "***Home***" controller and "***Index***" action are default values. Now, there are certain application level settings which are predefined in "***Global.asax***" file. This file is really interesting file in ASP.Net. It derives the application. Below is the sample code of this **Global.asax** file.

```
public class MvcApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();

        WebApiConfig.Register(GlobalConfiguration.Configuration);
        FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
        RouteConfig.RegisterRoutes(RouteTable.Routes);
        BundleConfig.RegisterBundles(BundleTable.Bundles);
        AuthConfig.RegisterAuth();
    }
}
```

Here, you can see that this class is derived from http application and this allows us to drill into some application level events like "***Application_Start***". So, this method will be processed by ASP.NET, before you process your 1st http request. So, it means when your application starts running, the code here will execute one time before any

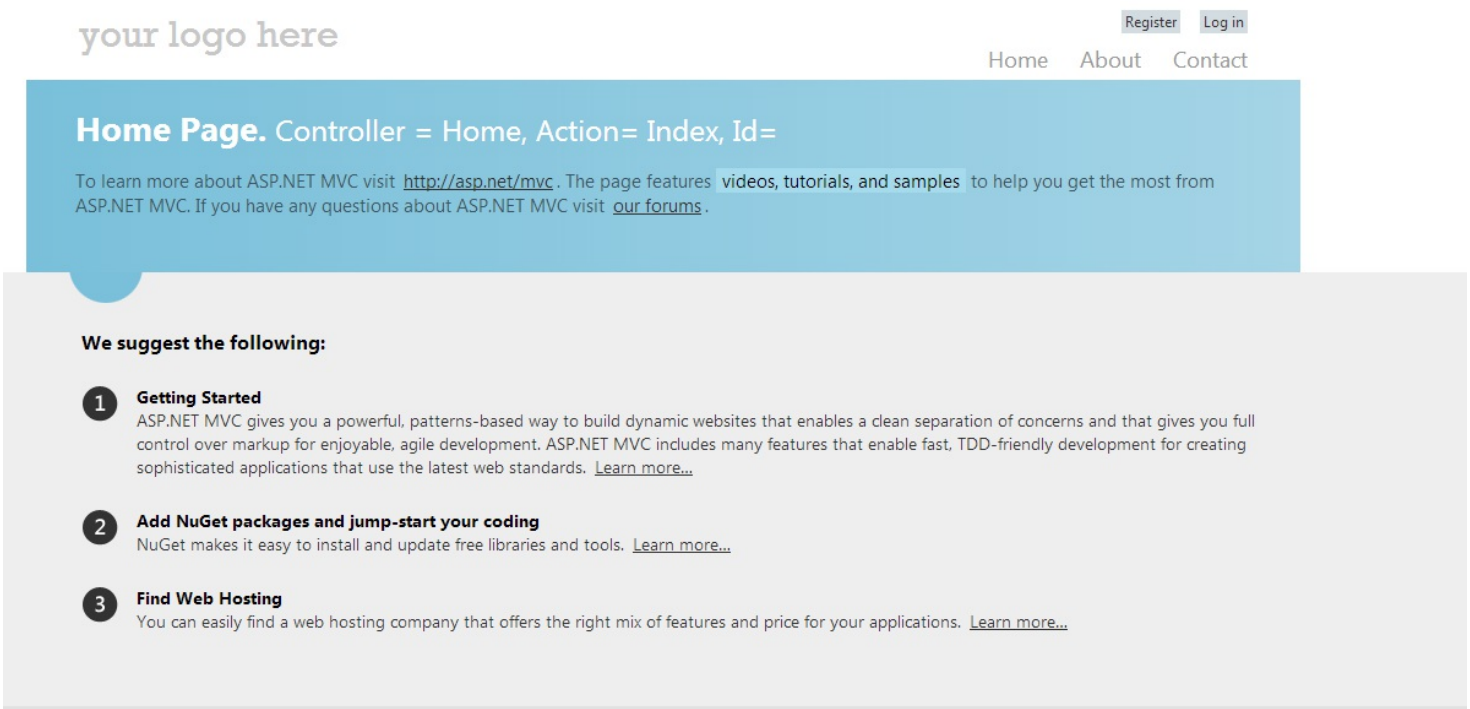
controllers start executing. So, this is the place where in we put some configuration related changes for example Routing Configuration. The route configuration is done by "RouteConfig.RegisterRoutes(RouteTabl
So, here we are going to pass the routing table also known as "***Global Routing Table***", values which contain all the routes for the entire application. This table will be empty initially, but when we call routes, it will add entries in the routing table. Now, let's inspect the values inside the route collection.

```
public ActionResult Index()
        {
            var Controller = RouteData.Values["controller"];
            var Action = RouteData.Values["action"];
            var Id = RouteData.Values["id"];

            string Output = string.Format("Controller = {0}, Action= {1}, Id= {2}",
Controller, Action, Id);
            ViewBag.Message = Output;

            return View();
        }
```

Now, when I build the solution and run the app, it will produce me the below result.



Now, if you see the above screen shot, it

presented the Controller's name and Action name. However, if you see that Id field is blank here, because while running the app, I have not invoked the app with any parameters, hence Id field is blank. However, there are other ways as well to grab these values. We'll see the same in some time.

Now, in this application I would like segregate my custom route with the default Route. So, let's go to "***RouteConfig.cs***" file under "***App_Start***" folder. Here, am going to write my custom route. However, before jumping directly to write a new route, let's 1st understand, why it is required to write another one when already default

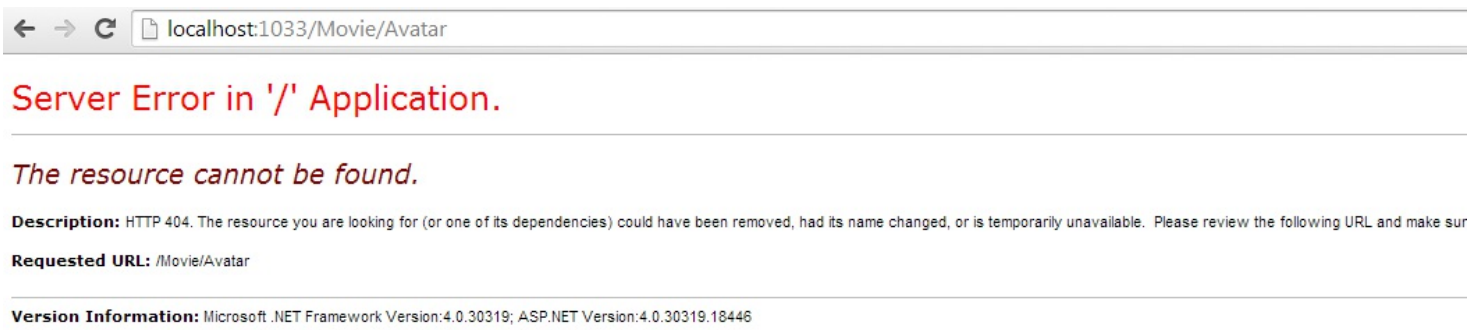
route is there. Consider a scenario where in any user comes in and search for a movie say "***Movie/Avatar***". In this case, Avatar will be treated as parameter name not like action method, so in this kind of scenario our default route won't work significantly.

So, in order to fix the same I'll write below route. Also, route placement is equally important. Because "***Default Route***" is very greedy, it matches almost any pattern. Note that route works on the principle first match wins, so, it's always suggested to place your new route in front of the default route. Also, as per MVC convention route name has to be unique. So, two routes cannot have

the same name.

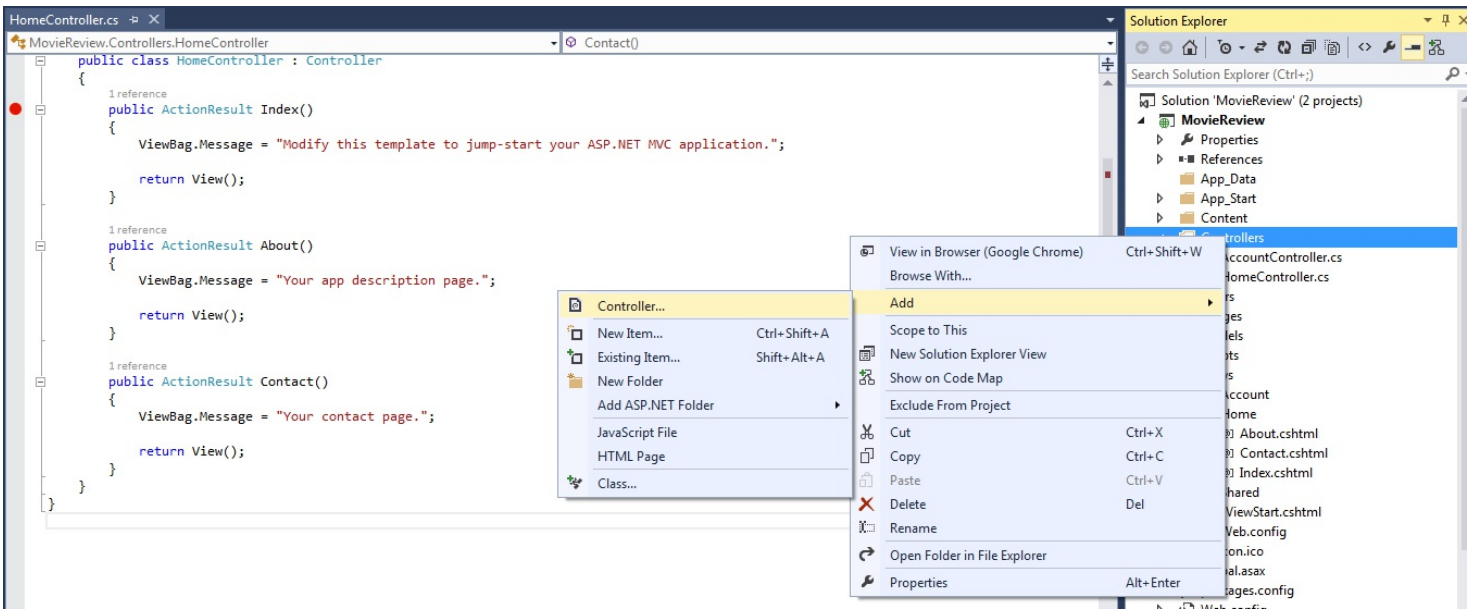
```
//Movie/Avatar
            routes.MapRoute("Movie",
                "Movie/{name}",
                new { controller = "Movie", action = "Search", name = "" });
```

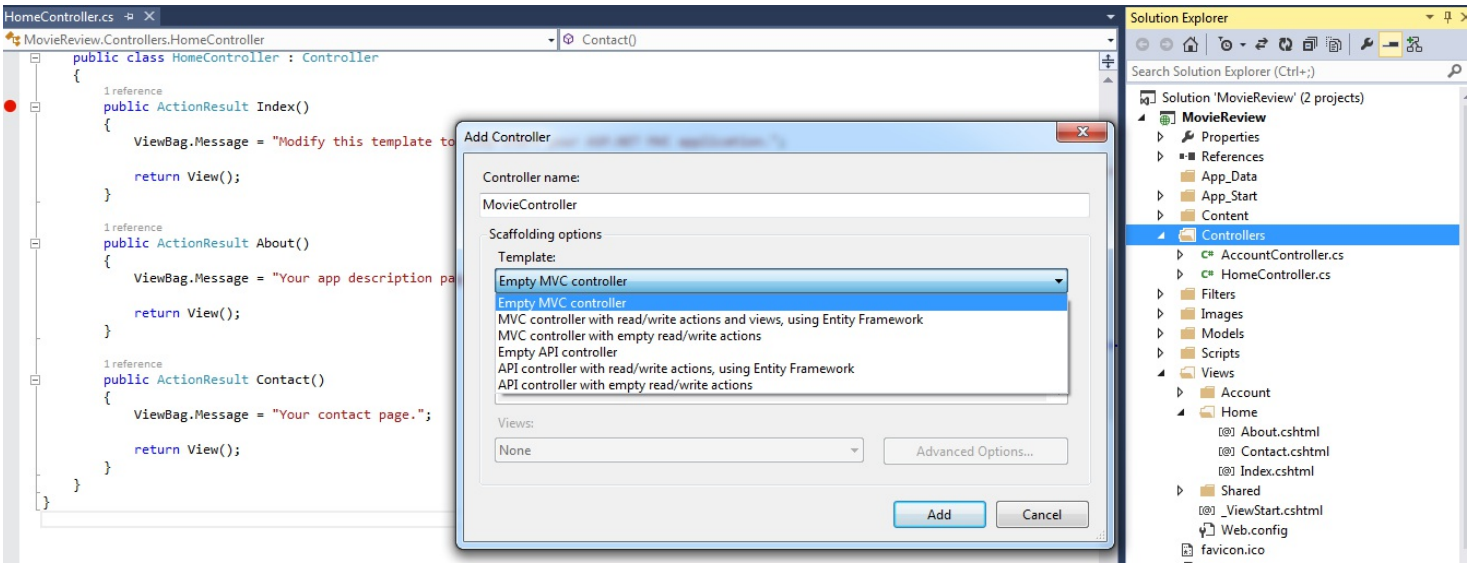
Now, with this new route in place, let's build the app and try to navigate (*http://localhost:1033/Movie/Avatar*) to the above mentioned URL and see the result.



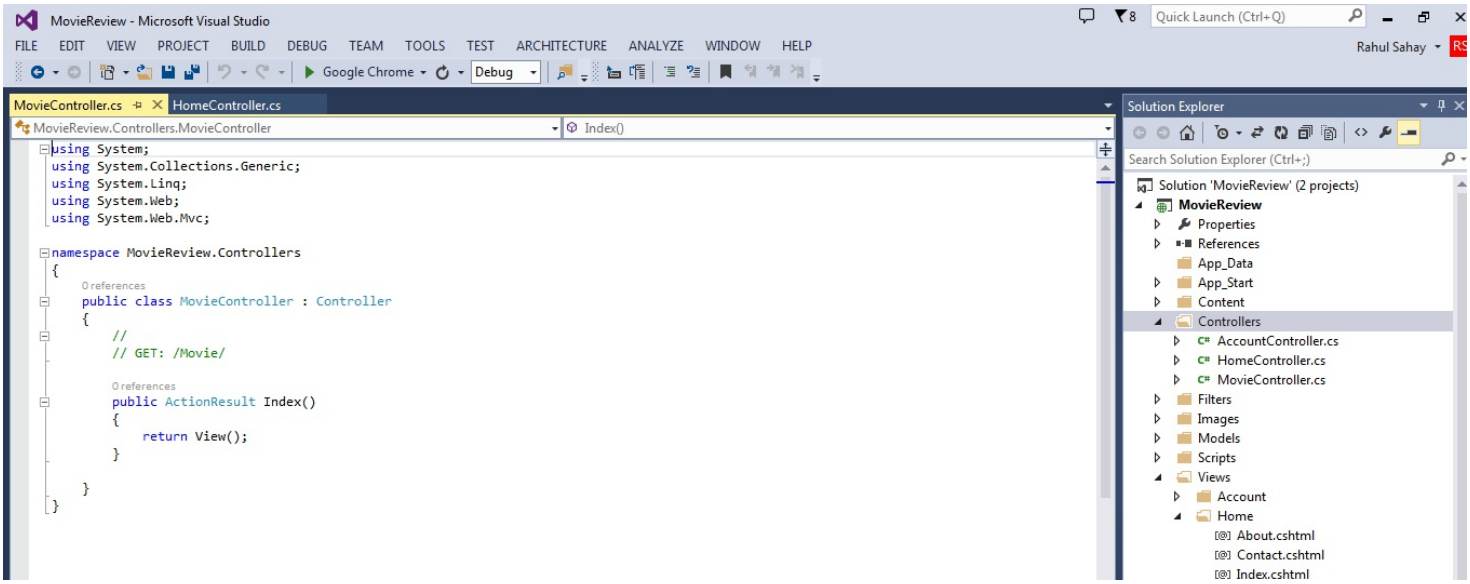
Well, the above result was obvious; because I told to go and find out the "***Movie Controller***", but since I have not

created the movie controller yet, hence it presented me the 404 error. Now, let me go ahead and create my 1st controller. Right-Click the controller's folder and say add new controller and then give the name as Movie Controller





Now, as you can see in the above screen shot I have added Movie Controller as an empty MVC controller. I took this liberty to write the entire code right from the scratch in front of you. So, the code for the same looks like as shown below. However, in the coming chapters we'll learn the usage of other scaffolding techniques listed in the dropdown.



Now, here in the above screen shot as you can see that it created the default action method with the name "***Index***". However, I have designed the route in such a way that it will look for "***Search***" action by default. Hence, let me go ahead and rename the Index action to Search action. Also, instead of returning view for now I'll just return Content so that I don't have to create view necessarily and also my purpose is served.

MovieReview.Controllers.MovieController ▾ Search()

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace MovieReview.Controllers
{
    0 references
    public class MovieController : Controller
    {
        //
        // GET: /Movie/

        0 references
        public ActionResult Search()
        {
            return Content("Returned from the Search Action from the Movie Controller");
        }

    }
}
```

So, now when I build the page and refresh the same, it will simply return me content on the browser.



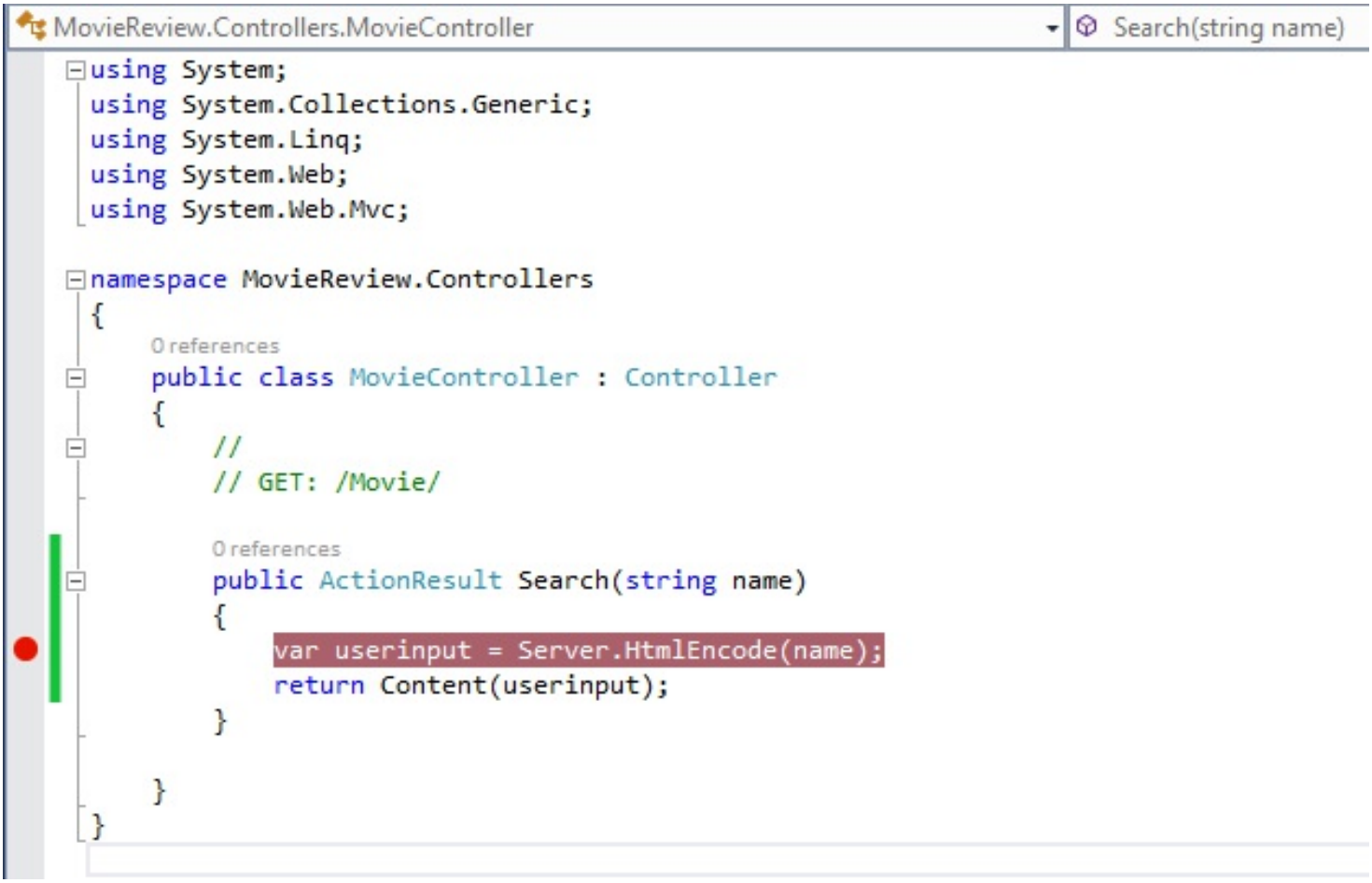
Returned from the Search Action from the Movie Controller

Now, let's think on how to fetch the incoming value from the browser when

parameter is getting passed in. However, there are couples of ways to fetch the route data from the URL, one way which I explained earlier in this chapter using the "***RouteData***" data structure and grab the value for the same. However, MVC framework also provided simple way to address this problem and that is by using the parameter inside the action method.

So, if you add a parameter to this method, what MVC framework will do, it will go and look for the value that matches the parameter name and it return it to you. Basically, it will apply all the permutation and combination to find out the parameter value for you, it will search in the Route data; it will also

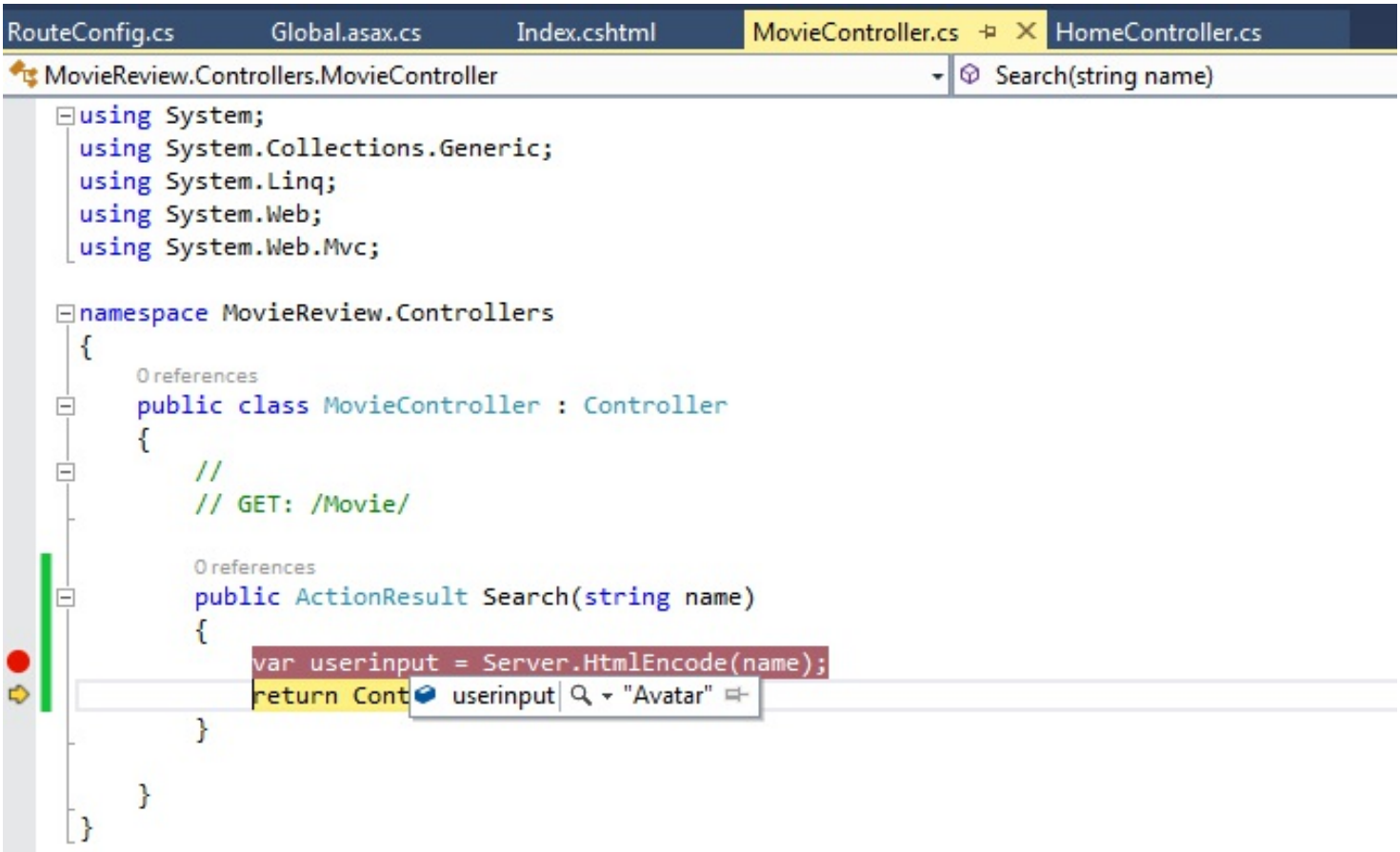
inspect the query string and in the form posted value. So, in this case "***/Movie/Avatar***", MVC framework will decide that I need a parameter with "***UserInput***", and then this will get picked from the URL and that will get automatically passed in to me.



Here, as you can see in the above screenshot in order to sanitize the user input I have used "***Server.HtmlEncode***", this will simply convert any kind of malicious script in plain text. If I would have been returning the same in view, it could have been automatically taken care for me. But, since I am returning the same with content view, hence I need to return the sanitized input to prevent any kind of scripting attack. Now, when I build and refresh the same, it will present me the below output.



You can also inspect the value for the same as well.



However, in the current scenario if I remove the parameter name and then invoke the URL that will work fine as shown below.



Reason for this is the default value of the route, in the route I have mentioned "***name=""***" which means if nothing is getting passed in, then simply pass in the empty value as default parameter value. However, if I remove the name parameter from the route and then build and refresh the page, then it will result me error as shown below.

```
//Movie/Avatar
routes.MapRoute("Movie",
    "Movie/{name}",
    new { controller = "Movie", action = "Search" });
```

Server Error in '/' Application.

The resource cannot be found.

Description: HTTP 404. The resource you are looking for (or one of its dependencies) could have been removed, had its name changed, or is temporarily unavailable. Please review the following URL and make sure that it is spelled correctly.

Requested URL: /movie

Version Information: Microsoft .NET Framework Version:4.0.30319; ASP.NET Version:4.0.30319.18446

So, in order to fix this I can go ahead and allow optional parameters for the parameter value as shown below.

```
//Movie/Avatar
routes.MapRoute("Movie",
    "Movie/{name}",
    new { controller = "Movie", action = "Search",name=UrlParameter.Optional });
```

What this is doing that it just ensuring that even if the parameter didn't mention in the url, it's ok it will not result 404 error, it will just print out the empty result. Also, in order to prove the point that MVC framework looking for the parameter values at different locations; let me try one example with query string.



Avatar

We can also provide certain default values if nothing is passed in as parameter in the URL. In that case, it will simply return the default value on the screen as shown below.

MovieReview.Controllers.MovieController | Search(string n

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace MovieReview.Controllers
{
    0 references
    public class MovieController : Controller
    {
        //
        // GET: /Movie/

        0 references
        public ActionResult Search(string name="Empty Value")
        {
            var userinput = Server.HtmlEncode(name);
            return Content(userinput);
        }

    }
}
```



Empty Value

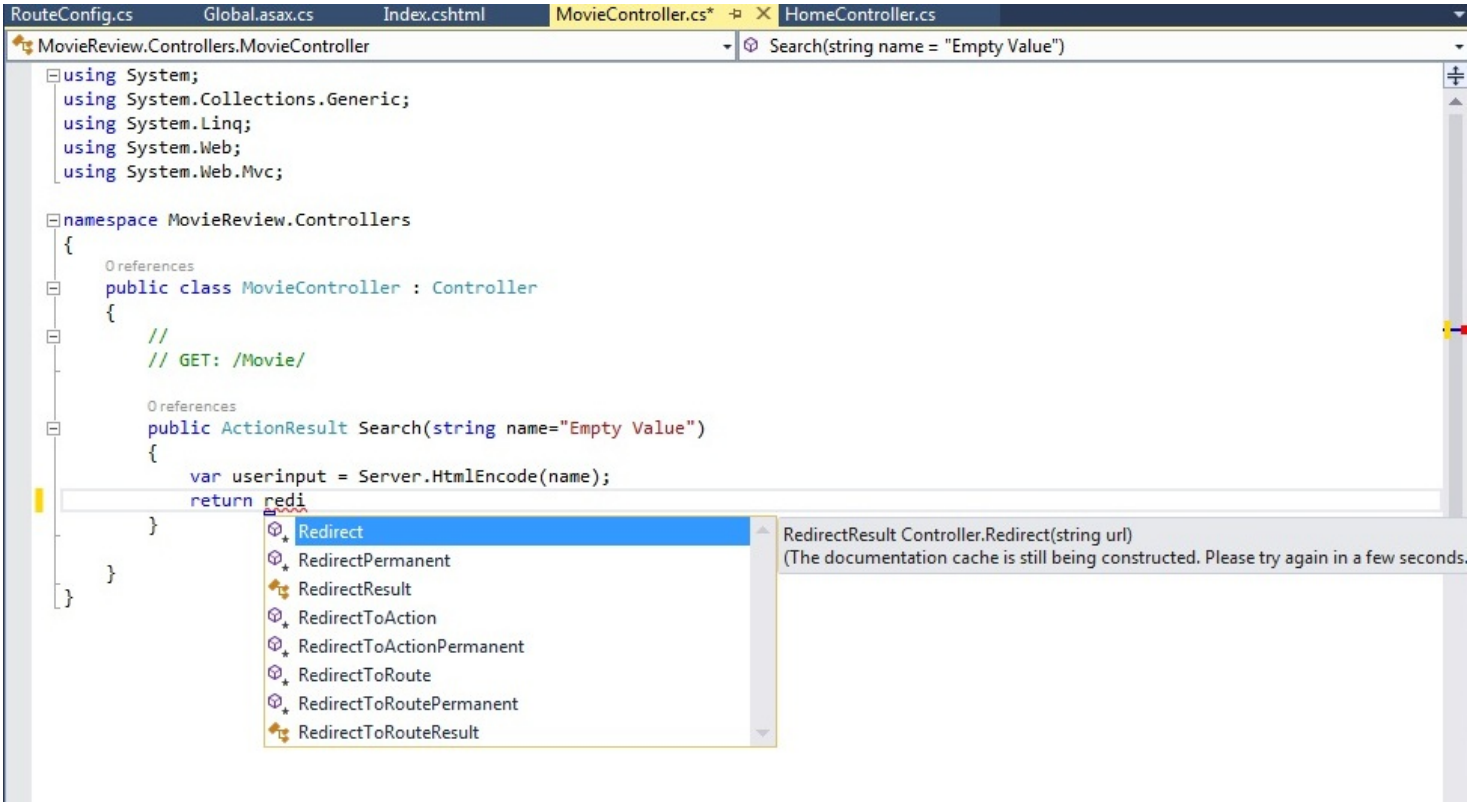
Action Results:-

Now, let's move forward and see different kinds of Action result. So, Actions basically returns different type of Action Results. I have put one brief table explaining different kind of action results and its behavior. We'll check them in detail with example shortly.

Name	Behavior
ContentResult	Returns a string
FileContentResult	Returns file content.
FilePathResult	Returns file content.
FileStreamResult	Returns file content.
EmptyResult	Returns nothing.
JavaScriptResult	Returns script for execution.
JsonResult	Returns JSON formatted data
RedirectToResult	Redirects to the specified URL.
HttpUnauthorizedResult	Returns 403 HTTP Status code.
RedirectToRouteResult	Redirect to different action/ different controller action.
ViewResult	Received as a response for view engine.
PartialViewResult	Received as a response for view engine.

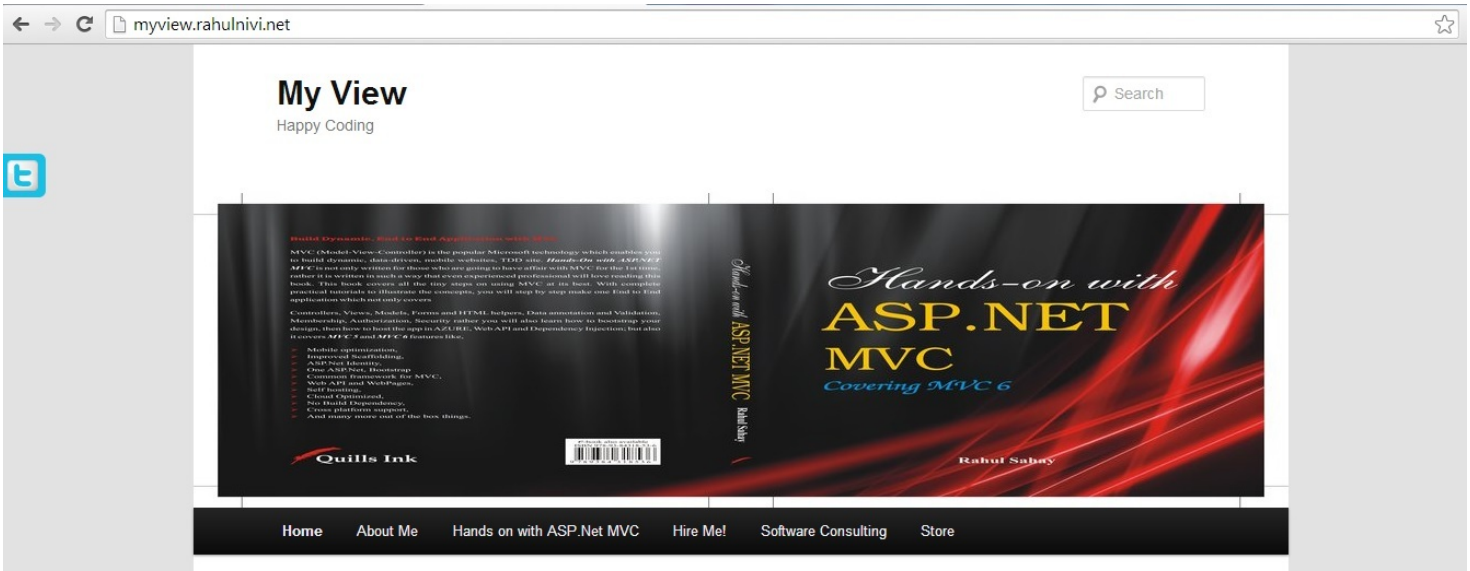
Here, the 1st column represents a type which is derived from the Action Result.

Now, let's do some demo on the above representations.

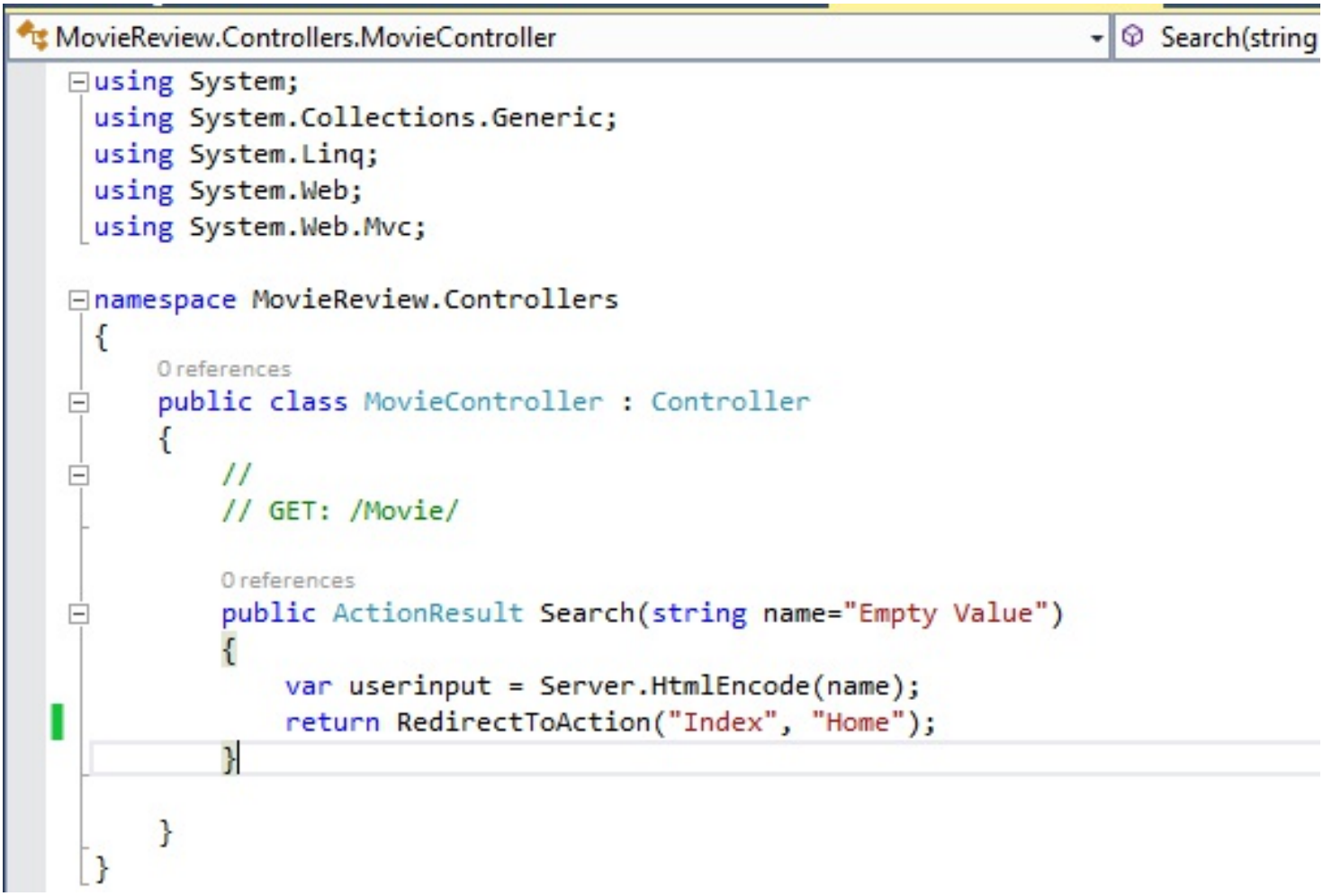


So, here in the above screen shot when you see the intellisense, you will see all different kind of redirect results. Now, let me consider one and redirect to my blog (***http://myview.rahulnivi.net***) . So, Now when I invoke this URL (***http://localhost:1033/movie***), it will

redirect me to my blog page as shown below.



I could also redirect to actions for the same controller or even for a different controller. Also, I can pass in values to the controller's action method if any required by that particular action method call. Below, is the simple example for the same.



Now, when I build and refresh the page, it will redirect to the home page. However, in case if you need to build a URL, where in you need to pass few values, obviously you could do that MVC framework provides you an option to do that. Below, is the sample one explaining the same.

RouteConfig.cs | Global.asax.cs | Index.cshtml | MovieController.cs | HomeController.cs

MovieReview.Controllers.MovieController | Search(string name = "Empty Value")

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

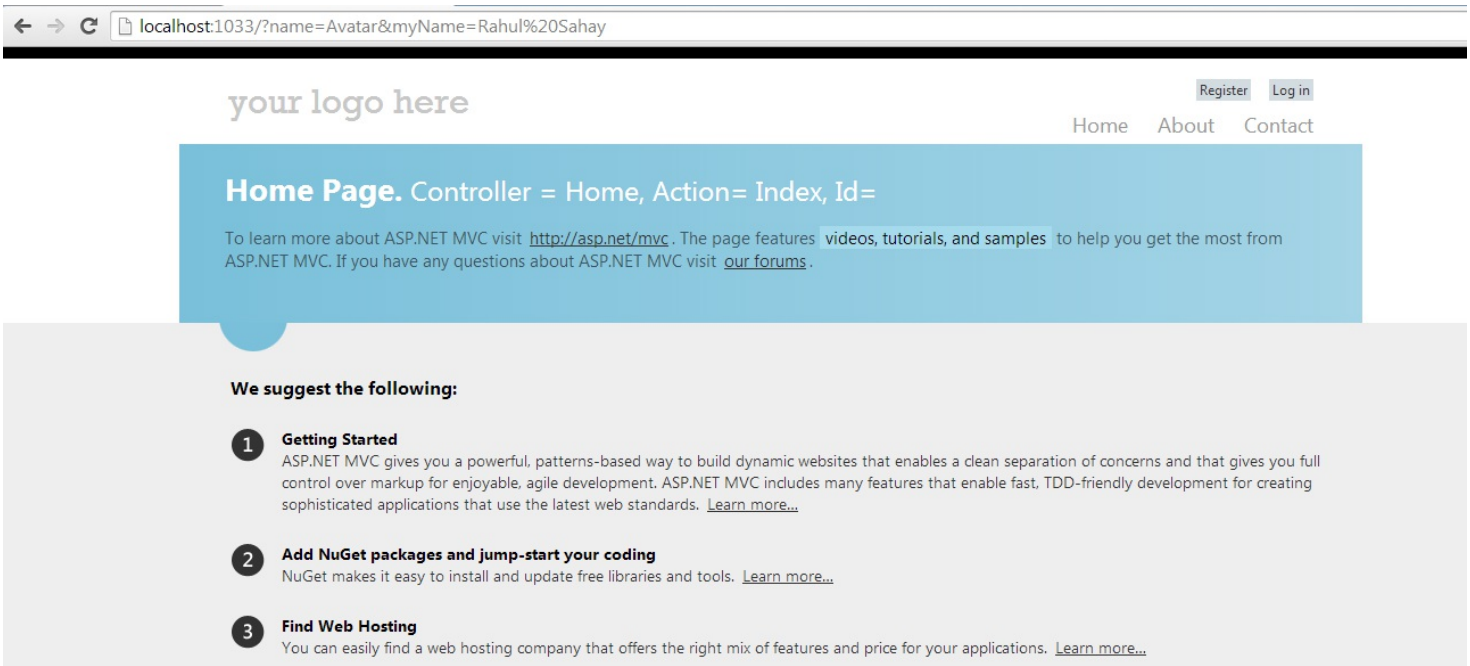
namespace MovieReview.Controllers
{
    0 references
    public class MovieController : Controller
    {
        //
        // GET: /Movie/

        0 references
        public ActionResult Search(string name="Empty Value")
        {
            var userinput = Server.HtmlEncode(name);
            return RedirectToAction("Index", "Home", new { name=userinput,myName = "Rahul Sahay"});
        }

    }
}
```

Now, here in the above screen shot, you could see that I have passed the values as anonymous type which is going to be present in the URL. So, there could be many scenarios in real world application, where in you need to pass in single or multiple values with action so that same values can be grabbed and utilized at that place. Now, when I build and invoke the URL

(http://localhost:1033/movie/Avatar) . This will produce me the below result with the modified URL (http://localhost:1033/?name=Avatar&myName=Rahul%20Sah having the values present in the query string.



However, we can also do something like directly redirecting to a route as shown below.

RouteConfig.cs | Global.asax.cs | Index.cshtml | MovieController.cs | HomeController.cs

MovieReview.Controllers.MovieController | Search(string name = "Empty Value")

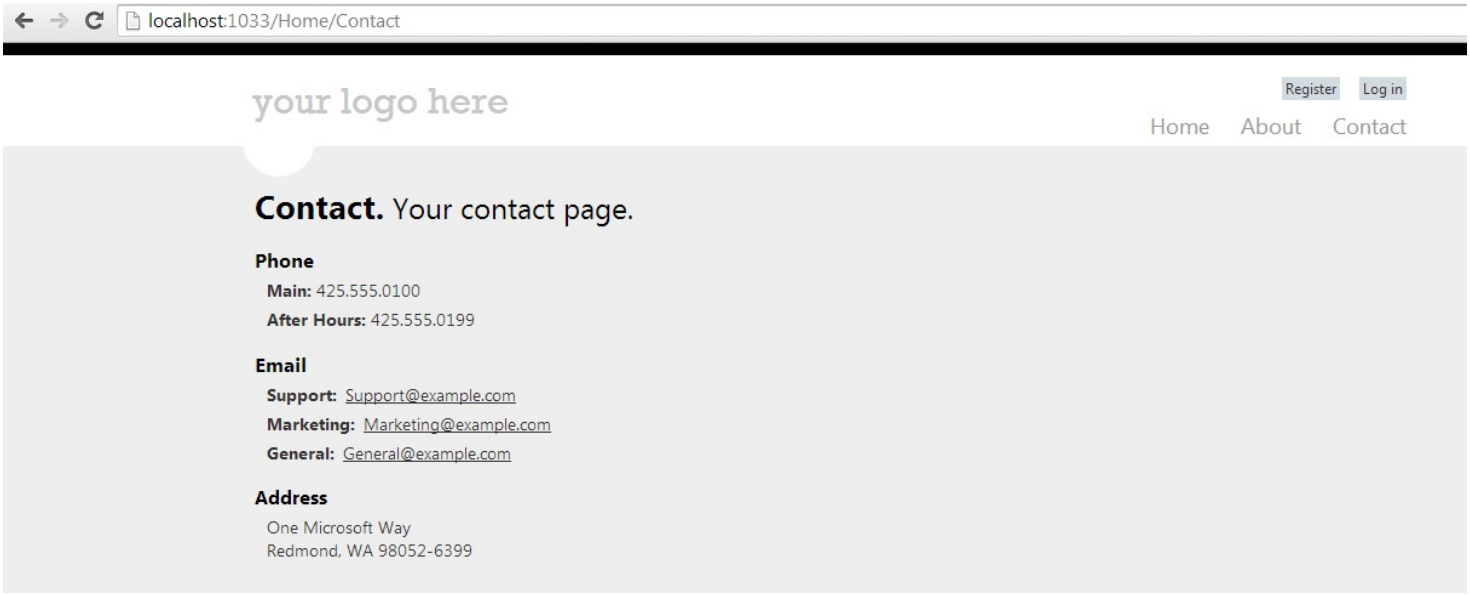
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace MovieReview.Controllers
{
    0 references
    public class MovieController : Controller
    {
        //
        // GET: /Movie/

        0 references
        public ActionResult Search(string name="Empty Value")
        {
            var userinput = Server.HtmlEncode(name);
            return RedirectToRoute("Default", new { controller = "Home", action = "Contact" });
        }

    }
}
```

Now, when you try building and invoke the url (*http://localhost:1033/movie/Avatar*), it will land up on the Contact page as shown below.



Now, let's try to work with files, let's suppose that you need to return a physical file stored in the file system. Then, how do you achieve the same. That is also very simple; remember the table which I have shown above, you can easily return any kind of physical files from the file system. Here, for explaining I am just picking one of the CSS files stored in themes folder.

RouteConfig.cs | Global.asax.cs | Index.cshtml | MovieController.cs | HomeController.cs

MovieReview.Controllers.MovieController | Search(string name = "Empty Value")

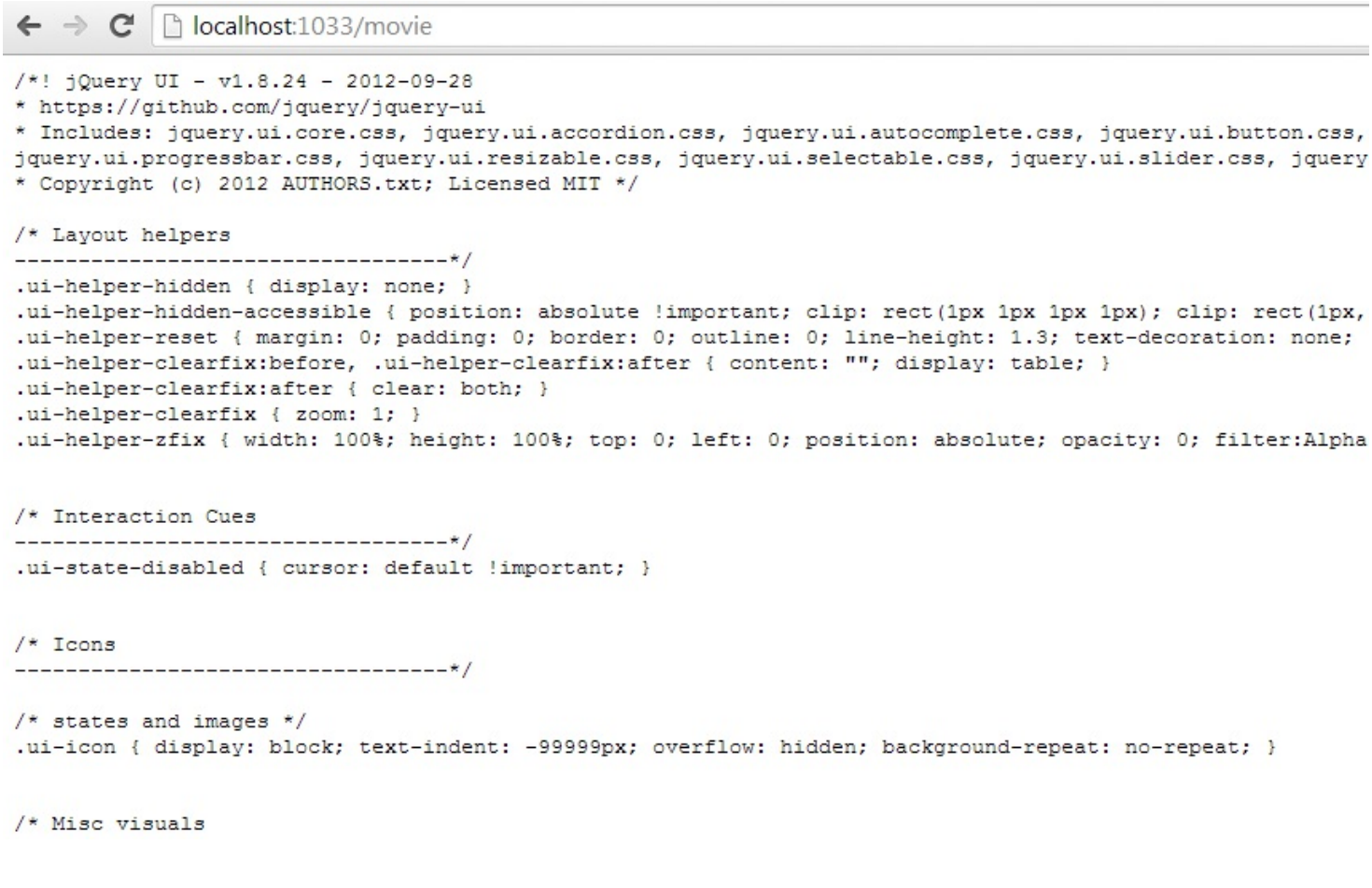
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace MovieReview.Controllers
{
    0 references
    public class MovieController : Controller
    {
        //
        // GET: /Movie/

        0 references
        public ActionResult Search(string name="Empty Value")
        {
            var userinput = Server.HtmlEncode(name);
            return File(Server.MapPath("~/Content/themes/base/jquery-ui.css"), "text/css");
        }

    }
}
```

In the above case, I have used map path method just to map the virtual path of the file system, basic C#; nothing rocket science. Now, when I build and refresh the page, then it will return me the physical file.



In similar context, you could return JSON as well.

RouteConfig.cs | Global.asax.cs | Index.cshtml | MovieController.cs | HomeController.cs

MovieReview.Controllers.MovieController | Search(string name = "Empty Value")

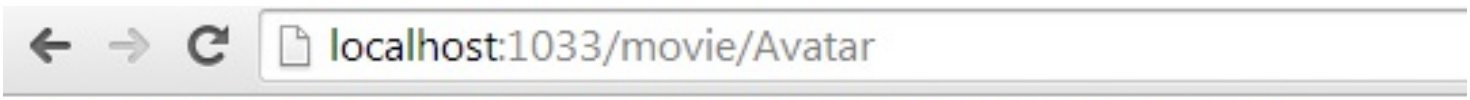
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace MovieReview.Controllers
{
    0 references
    public class MovieController : Controller
    {
        //
        // GET: /Movie/

        0 references
        public ActionResult Search(string name="Empty Value")
        {
            var userinput = Server.HtmlEncode(name);
            return Json(new { Fname = "Rahul", Lname = "Sahay", MovieSelected = userinput }, JsonRequestBehavior.AllowGet);
        }

    }
}
```

Now, when you build and refresh it will produce you the JSON result as shown below.




```
{"Fname":"Rahul","Lname":"Sahay","MovieSelected":"Avatar"}
```


So, this was few of the action results demo. Obviously you can go ahead and experiment with others as well.

Action Selectors:-
Action Names
Accept Verbs

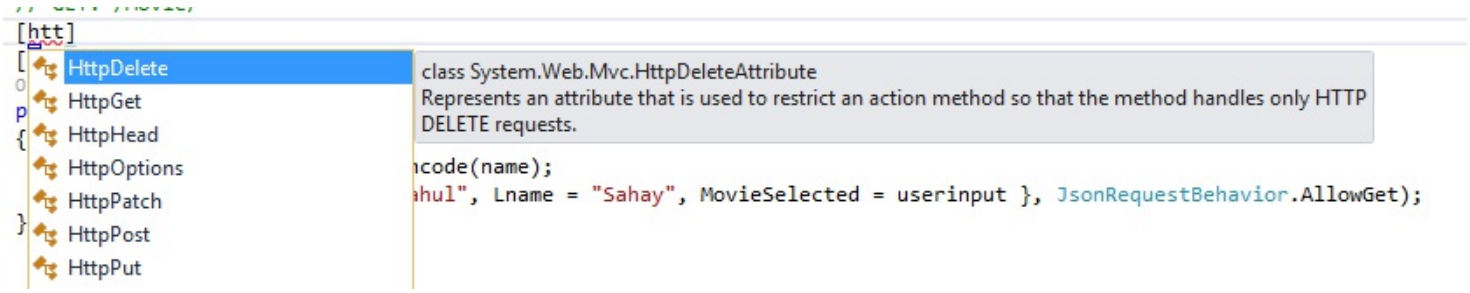
Action Selectors play crucial role when you are writing your action methods. These, selectors will decide the behavior of the method invocation based on the modified name given in front of the action method. So, basically this attribute we usually use when we want to alias the name of the action method. So, now this action cannot be reached with the name **Search**, it will now reach with the name "***Edit***". If you try reaching the action method with "*Search*", this will return a 404 as routing engine won't

be able to resolve the request. This approach we also take to make more user friendly Action name.

```
[ActionName("Edit")]
0 references
public ActionResult Search(string name = "Empty Value")
{
    var userinput = Server.HtmlEncode(name);
    return Json(new { Fname = "Rahul", Lname = "Sahay", MovieSelected = userinput }, JsonRequestBehavior.AllowGet);
}
```

Now, Accept Verbs are terminology which will take care of HTTP Verbs. So, you could basically restrict an action method based on the Http Verbs like you can say, that this action method is only reachable when **HTTP-POST/HTTP-GET** request is being issued. Below I have mentioned the snapshot of all the HTTP Accept verbs via Visual Studio intellisense. So, you can obviously play around these verbs based on your

requirement.



Now, in order to explain the distinction between these Accept Verbs, let me make a small change in the action method. Below, if you see I have annotated the action method with HttpPost method.

RouteConfig.cs | Global.asax.cs | Index.cshtml | MovieController.cs | HomeController.cs

MovieReview.Controllers.MovieController — Search(string name = "Empty Value")

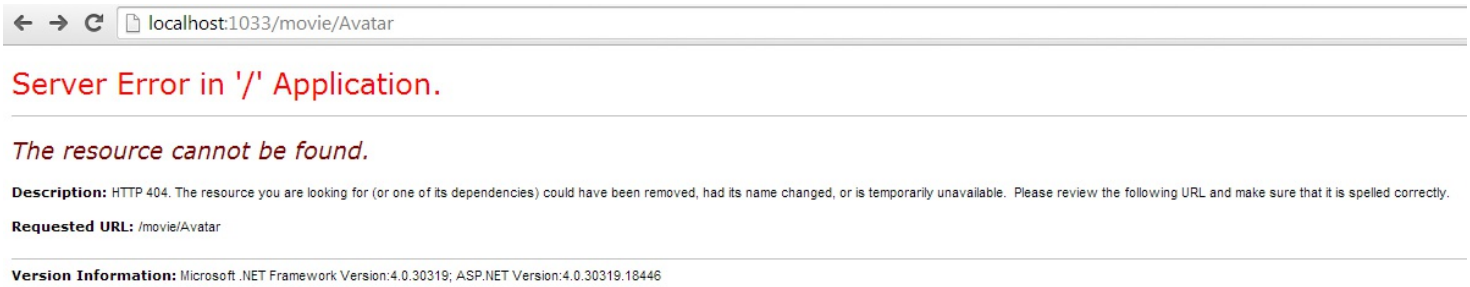
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace MovieReview.Controllers
{
    0 references
    public class MovieController : Controller
    {
        //
        // GET: /Movie/

        [HttpPost]
        0 references
        public ActionResult Search(string name="Empty Value")
        {
            var userinput = Server.HtmlEncode(name);
            return Json(new { Fname = "Rahul", Lname = "Sahay", MovieSelected = userinput }, JsonRequestBehavior.AllowGet);
        }

    }
}
```

So, now when I build and refresh the page, it will return me 404 error, because MVC framework could find an action method which is there in the controller equivalent to HttpGet request.



Now, in order to fix this scenario we can

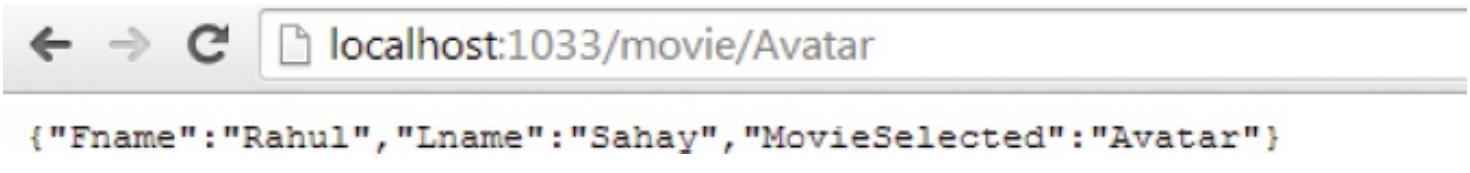
have the same action method but this time annotated with the HttpGet as shown below.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace MovieReview.Controllers
{
    0 references
    public class MovieController : Controller
    {
        //
        // GET: /Movie/
        [HttpGet]
        0 references
        public ActionResult Search(string name = "Empty Value")
        {
            var userinput = Server.HtmlEncode(name);
            return Json(new { Fname = "Rahul", Lname = "Sahay", MovieSelected = userinput }, JsonRequestBehavior.AllowGet);
        }


    }
}
```

Now, when I build and refresh, I should get the desired result back.



However, let's understand why this is important to use the Accept Verbs. For a

moment let's assume you have another action with the same name as shown below.

```
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

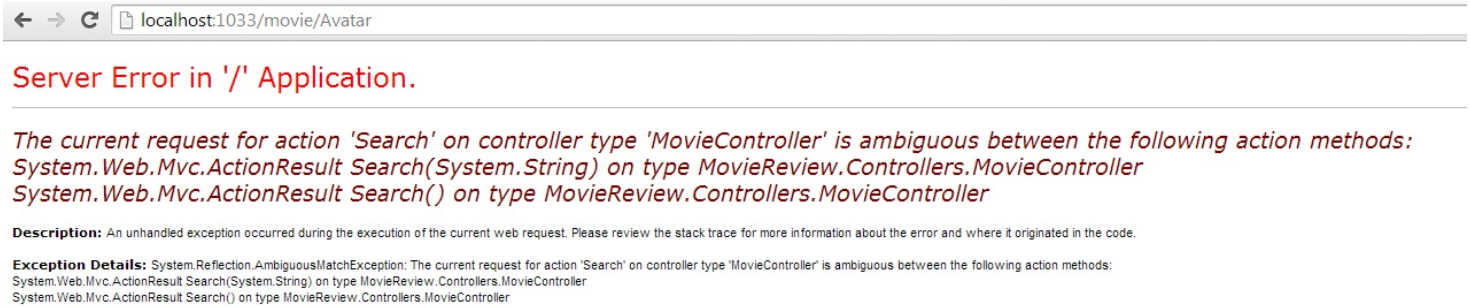
namespace MovieReview.Controllers
{
    0 references
    public class MovieController : Controller
    {
        //
        // GET: /Movie/

        0 references
        public ActionResult Search(string name = "Empty Value")
        {
            var userinput = Server.HtmlEncode(name);
            return Json(new { Fname = "Rahul", Lname = "Sahay", MovieSelected = userinput }, JsonRequestBehavior.AllowGet);
        }

        0 references
        public ActionResult Search()
        {
            return Json(new { Fname = "Rahul", Lname = "Sahay" }, JsonRequestBehavior.AllowGet);
        }

    }
}
```

Now, when you build the same it will build successfully. However, when you refresh the page, it will result the below error.



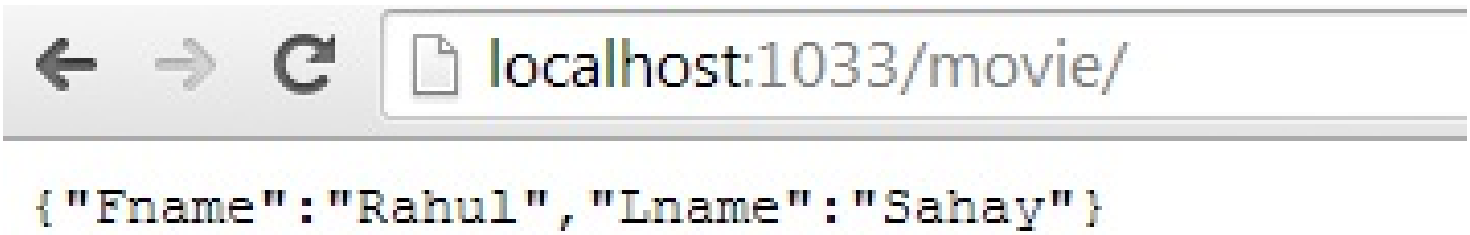
Now, when you look at the error, this is very obvious as it says MVC framework unable to figure out which action method should be picked up for the get request. So, in order to rescue us from these kinds of scenarios, Accept Verbs come in action. Now, in order to get rid of the error, just annotate the same with different Accept verbs as shown below.

```
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace MovieReview.Controllers
{
    0 references
    public class MovieController : Controller
    {
        //
        // GET: /Movie/
        [HttpPost]
        0 references
        public ActionResult Search(string name = "Empty Value")
        {
            var userinput = Server.HtmlEncode(name);
            return Json(new { Fname = "Rahul", Lname = "Sahay", MovieSelected = userinput }, JsonRequestBehavior.AllowGet);
        }
        [HttpGet]
        0 references
        public ActionResult Search()
        {
            return Json(new { Fname = "Rahul", Lname = "Sahay" }, JsonRequestBehavior.AllowGet);
        }

    }
}
```

Now, when I do build and refresh, error should have gone now.



So, this was the basic demo for Accept Verbs. Now, let's go ahead and see Action Filters.

Action Filters:-

Above we have discussed about Action Selectors. Apart from action selectors, there is one more topic which is equally important, that is Action Filter. Below, I have mentioned their usage in brief.

Name	Behavior
Authorize	Actions are restricted to allow only authorized users.
ValidateInput	Allow all kind of input.
OutputCache	Caches controller's output.
HandleError	User can specify custom views for handling different kind of exceptions.
ValidateAntiForgeryToken	Prevent Cross site scripting.

So, basically by definition filters are the one which apply pre/post processing logic to a controller action. We'll do deep discussion about these filters in coming chapters.

Summary:-

In this chapter, we discussed controller's job and also that how controller controls the nerve of the application. We see many things like basics of controller, did some debugging with the controllers, discussed routing in brief. We also discussed Action Results, Action Selectors and Action Filters. I have covered these topics here in controller to keep the things really simple; because as we progresses with different chapters, these topics will be used immensely in designing the application.

Chapter 3: Views

WHAT DO you find in this CHAPTER?

⇒ Introduction

⇒ Views

⇒ Syntax of Razor Views

⇒ How to Prevent Cross Site Scripting Attack

⇒ More on Razor Syntax

⇒ Layout Views

⇒ HTML Helpers

⇒ HTML.ValidationSumma

⇒ Some more HTML Helpers

⇒ Partial Views

⇒ Viewbag and Viewdata

⇒ Summary

Introduction:-

So, basically Razor View engines are responsible for creating HTML Outputs. Razor views are nothing but files ending with "***.cshtml***" extension. So, if I have to summarize Razor in one step I would be doing like shown below

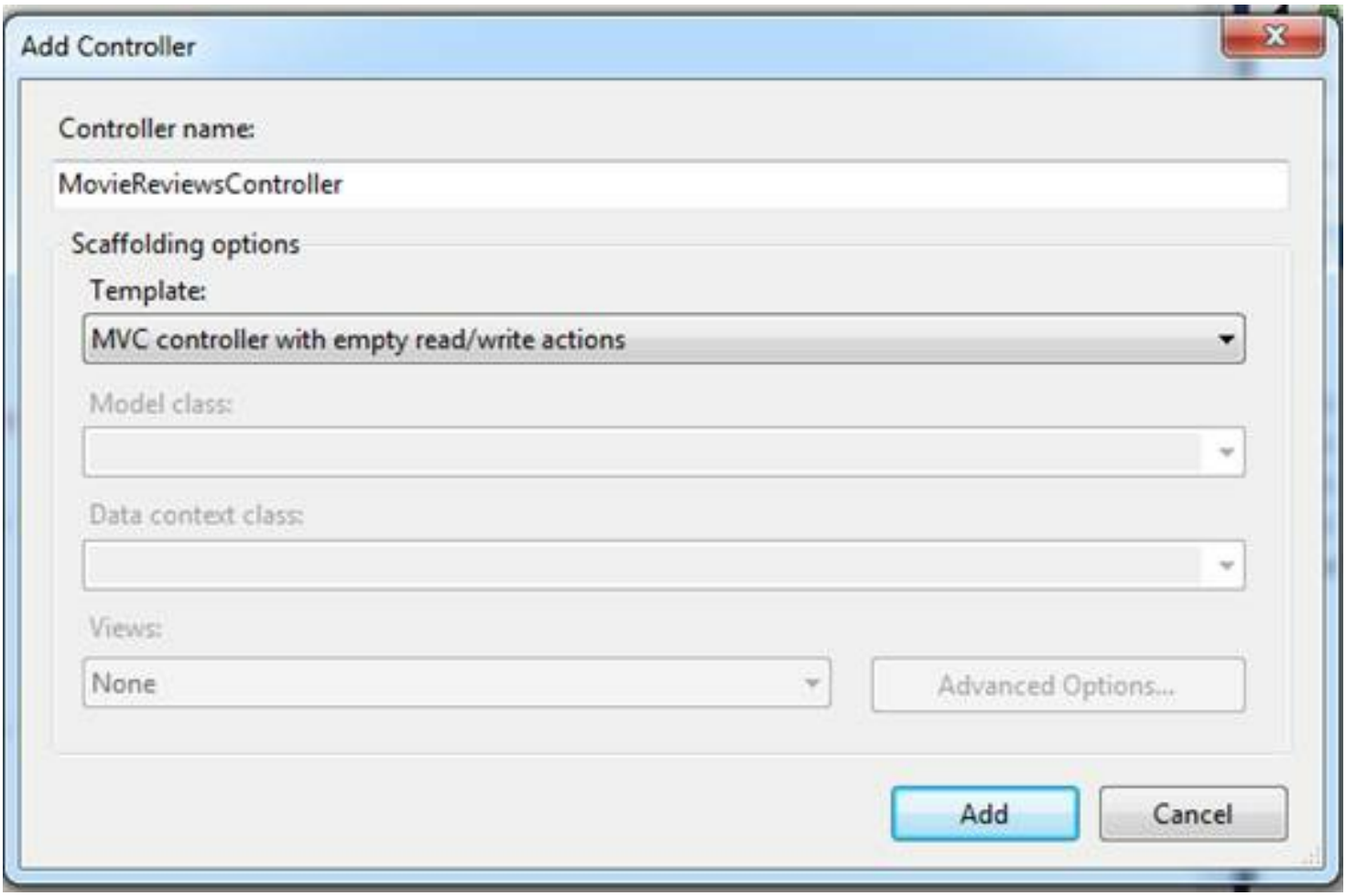
"***Static Template + Model Data = Produced HTML*** "

So, the above formula says that you have a template in which with the help of C# code you are going to inject data pieces somewhere in between the markup. And the final output which renders is going to be the combination of static HTML plus

the data fetched from the database or from any other remote resources. Now, let's look at the same in Visual Studio, to make the above description more robust.

Views:-

So, imagine a scenario that at a section of page, I would like to have latest reviews pulled in from the database. For now, we won't be working with database; we'll fetch from database in the coming chapters. But, for now we'll use some hard coded values in order to prove my point. So, now let me go ahead and create one controller with the name "***MovieReviews***".



Now, this time I have selected a different option from scaffolding dropdown. Now, this scaffolding template will create an **Index** method with default action for controller. However, this will also list other methods like **Edit/Delete/Create** a review as well.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace MovieReview.Controllers
{
    public class MovieReviewsController : Controller
    {
        //
        // GET: /MovieReviews/

        public ActionResult Index()
        {
            return View();
        }

        //
        // GET: /MovieReviews/Details/5

        public ActionResult Details(int id)
        {
            return View();
        }

        //
        // GET: /MovieReviews/Create

        public ActionResult Create()
        {
            return View();
        }

        //
        // POST: /MovieReviews/Create

        [HttpPost]
        public ActionResult Create(FormCollection collection)
        {
            try
```

```
    {
        // TODO: Add insert logic here

        return RedirectToAction("Index");
    }
    catch
    {
        return View();
    }
}

//
// GET: /MovieReviews/Edit/5

public ActionResult Edit(int id)
{
    return View();
}

//
// POST: /MovieReviews/Edit/5

[HttpPost]
public ActionResult Edit(int id, FormCollection collection)
{
    try
    {
        // TODO: Add update logic here

        return RedirectToAction("Index");
    }
    catch
    {
        return View();
    }
}

//
// GET: /MovieReviews/Delete/5

public ActionResult Delete(int id)
{
    return View();
}
```

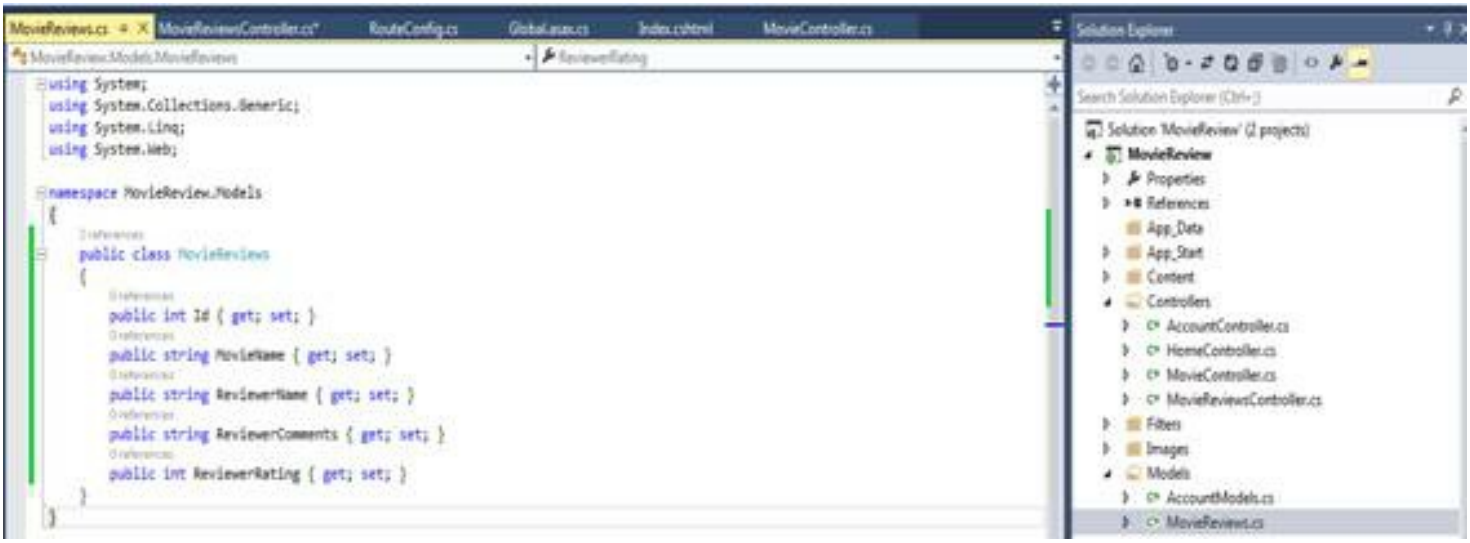
```
        //
        // POST: /MovieReviews/Delete/5

        [HttpPost]
        public ActionResult Delete(int id, FormCollection collection)
        {
            try
            {
                // TODO: Add delete logic here

                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }
    }
}
```

Now, here in the index action we are going to list the reviews. But, currently we are not having any model class with which this action can be wired up and produce the result. Hence, 1st I need to create one model class for this and

properties to that as shown below.



Now, in **MovieReviews** Controller I'll put some static data in there, so that same can be used to produce the result in views. Please note currently in this context I am not focusing on data access, am just focusing on the views.

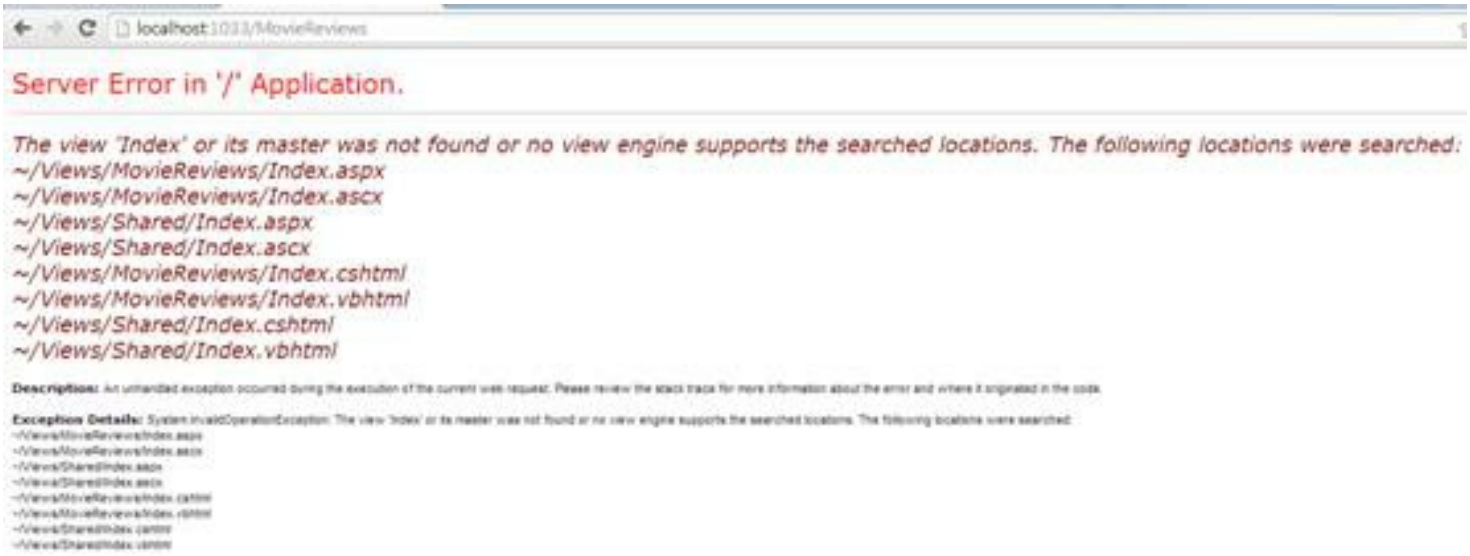
```
static List<MovieReviews> _movieReviews = new List<MovieReviews>
        {
            new MovieReviews{
```

```
        Id=1,
        MovieName="Avatar",
        ReviewerName="Rahul Sahay",
        ReviewerComments="Excellent",
        ReviewerRating=5
    },
    new MovieReviews{
        Id=2,
        MovieName="Titanic",
        ReviewerName="Rahul Sahay",
        ReviewerComments="Excellent",
        ReviewerRating=5
    },
    new MovieReviews{
        Id=3,
        MovieName="Die Another Day",
        ReviewerName="Rahul Sahay",
        ReviewerComments="Excellent",
        ReviewerRating=5
    }
};
```

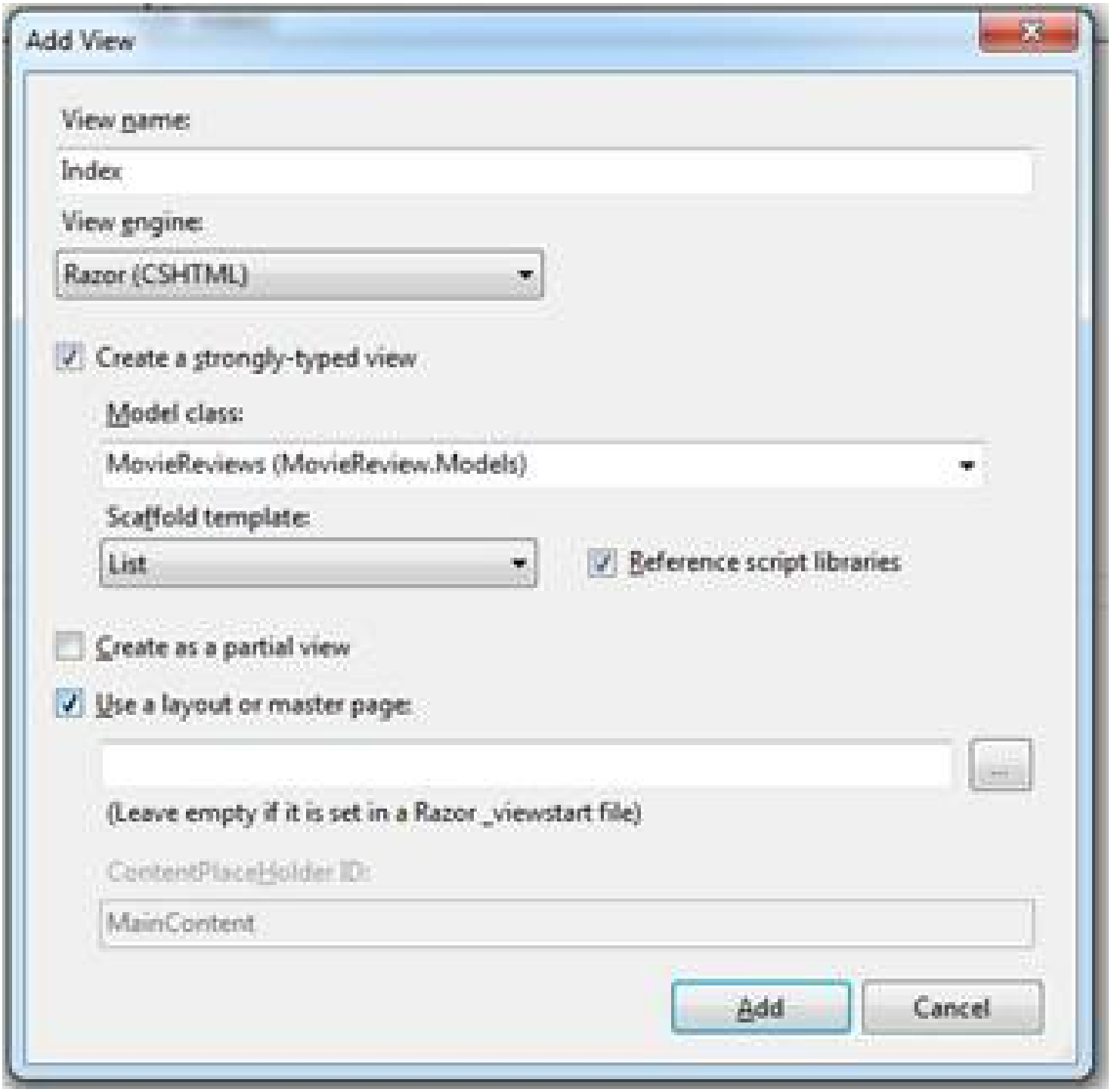
Now, I have a static list of data to work with the view. Now, I need to use the same in my index action. So, this is just basically a work around till we get the database up and running.

```
public ActionResult Index()
        {
            var model = from m in _movieReviews
                        orderby m.MovieName
                        select m;
            return View(model);
        }
```

So, now in the above snippet, I am just accessing the static data via LINQ query. Now, let's build the application and run the same, then it will present me the below error.



So, what we done here we asked the MVC runtime to render an **Index view** for MovieReviews, but MVC runtime couldn't find the one matching the above extensions at different locations. So, in order to fix the same, I need to create a view for it. Now, what I'll do is I'll right-click on the action method and say Add View and this time I'll create the strongly typed view as shown below in the dialog box.



Also, I have selected List from the Scaffolding template. What this scaffolding template will do, it will emit

the HTML markup in List fashion. Below, is the default code which is generated by the scaffolding template.

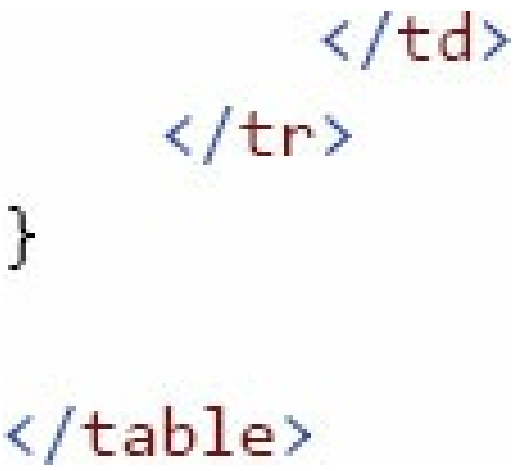
```
@model IEnumerable<MovieReview.Models.MovieReviews>
```

```
@{
    ViewBag.Title = "Index";
}

<h2>Index</h2>

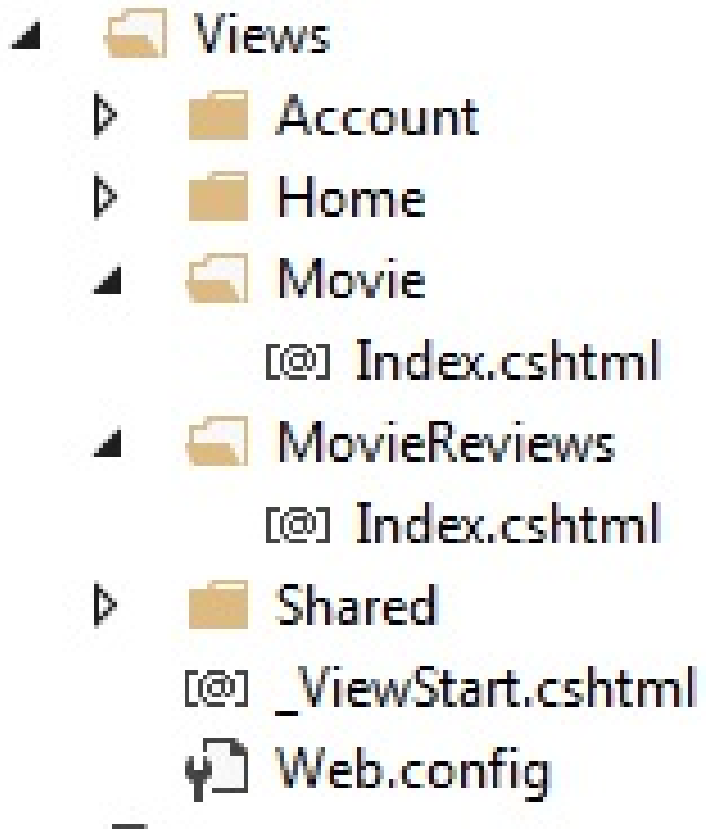
<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table>
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.MovieName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.ReviewerName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.ReviewerComments)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.ReviewerRating)
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.MovieName)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.ReviewerName)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.ReviewerComments)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.ReviewerRating)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.Id }) |
            @Html.ActionLink("Details", "Details", new { id=item.Id }) |
            @Html.ActionLink("Delete", "Delete", new { id=item.Id })
```



Scaffolding template is something which gives you boiler plate code means starting point, Visual Studio is smart enough to decide the view structure based on the model you created. It does the mapping accordingly. Now, in the above snippet, you could see that variable "***item***" is getting looped and producing individual item in the List format. Also, since the above code is generated by the visual studio, you could easily customize the template based on

your need. Visual Studio doesn't restrict you to change the styling or labeling of the template generated. Now, if you see the folder structure, one more folder has been created with the name **MovieReviews** under Views folder.

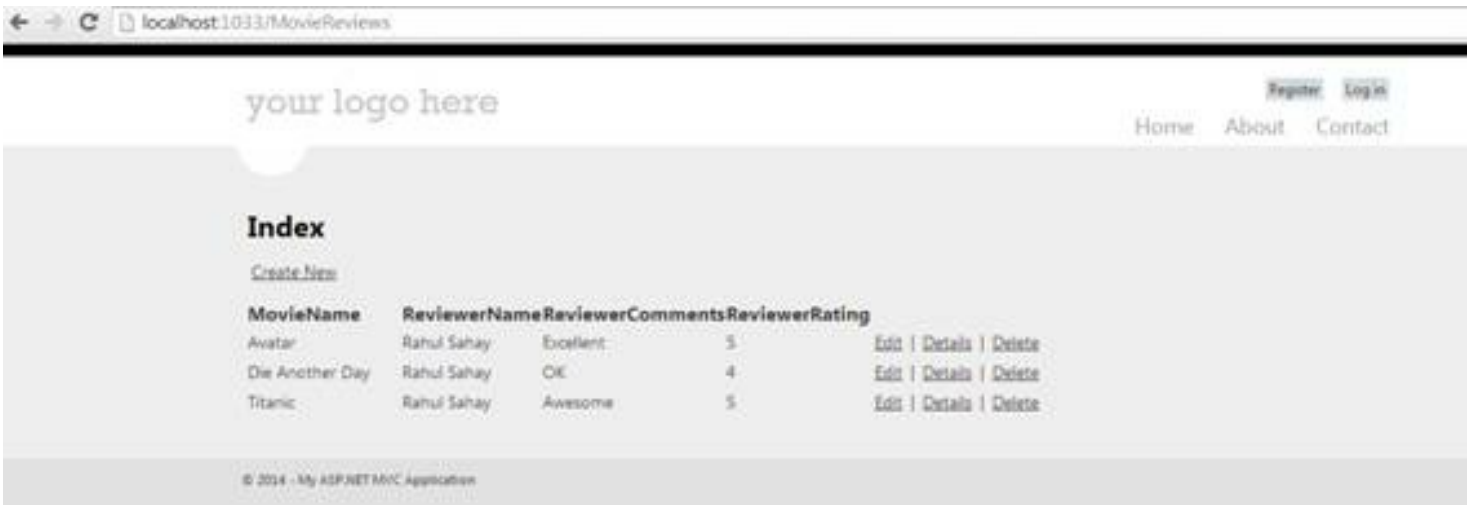


Now, at the top of the index view, when you inspect the code; you will find that

index view is strongly typed

```
@model IEnumerable<MovieReview.Models.MovieReviews>
```

With the **IEnumerable** of MovieReviews; it is going to return a collection of Movie reviews and obviously will build one table to display it. Now, when I build and refresh the page, I will be having the desired output.



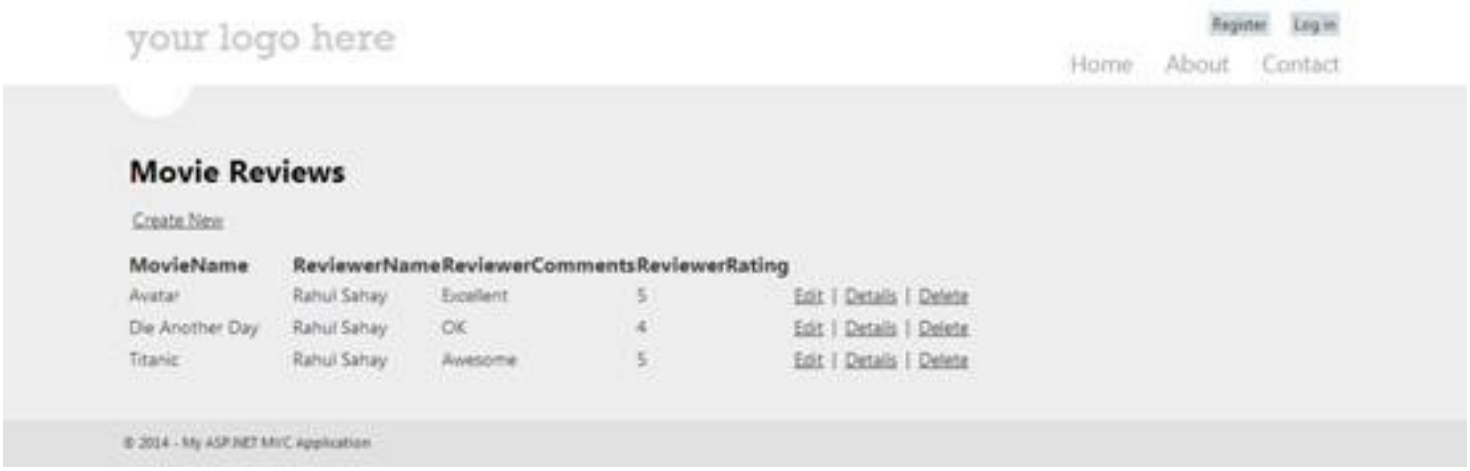
Syntax of Razor Views:-

Now, let's understand razor job. So, if you remember the View formula which I have introduced during the introduction of this chapter, Views are nothing but a combination of static template or string literals with data. In our case static template is HTML table presentation on the screen and injected data inside the table is the model which we passed in. Literal text over here is H2 tag which we have on the page or some sentence or word which you are writing on the page. You could obviously change the headers on the page; to make it more descriptive or meaningful like shown below.

```
<h2>Movie Reviews</h2>
```

Now, when I just save the file and

refresh the browser, this will take effect.

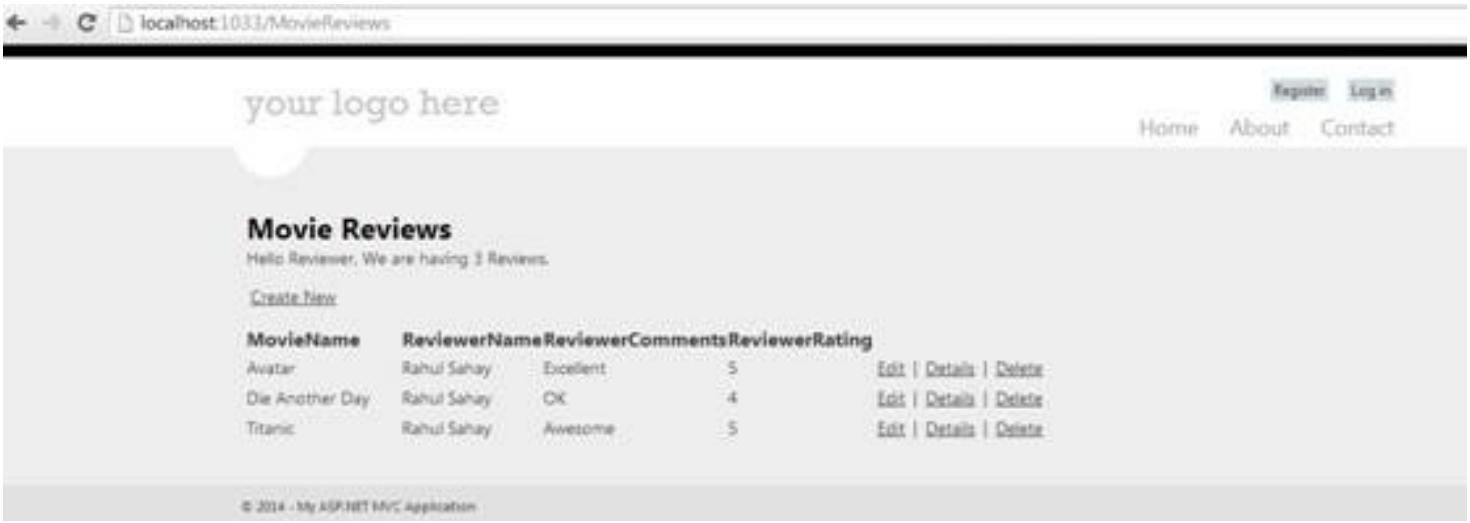


Now, as far as C# code is concerned, this must be prefixed with @ symbol. Now, let's suppose, in between the string literals, you would like to inject incoming data, then in that case we need to make use C# code to fetch the incoming model data as shown below.

```
Hello Reviewer, We are having @Model.Count() Reviews.
```

Now, after saving when I refresh the

page, it would produce me the below output.



So, with the above snippet in place, razor easily figured out ok that's a server side invocation, hence pulled in the value from the model supplied and hence presented on the browser. There are other things apart from the Model as well, which you could access from @symbol like **ViewBag**, **ViewData** etc. you can put the value in either of these

and can easily access the same in the razor view. We will talk about these **ViewBag** and **ViewData**later in this chapter.

How to prevent Cross Site Scripting Attack:-

Now, let me explain one more thing, if you remember from the controller's chapter, I said that if values are getting passed on to Razor views, those values are by default HTML encoded, you don't have to HTML Encode the data, but in case other than razor view, you need to be careful. Now, let's consider a scenario where a malicious user sitting

on the browser sends a request with script inside as shown below

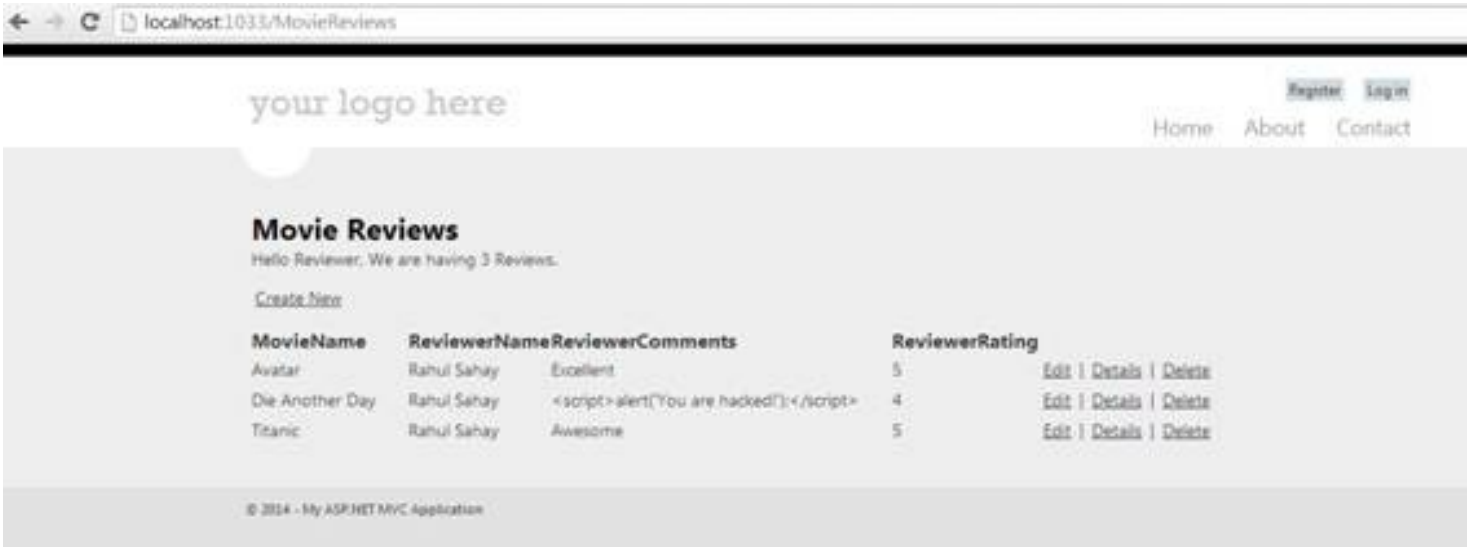
```
"<script>alert('You are hacked!');</script>"
```

In order to demonstrate the same, let me modify the incoming model data as shown below

```
static List<MovieReviews> _movieReviews = new List<MovieReviews>
        {
            new MovieReviews{
                Id=1,
                MovieName="Avatar",
                ReviewerName="Rahul Sahay",
                ReviewerComments="Excellent",
                ReviewerRating=5
            },
            new MovieReviews{
                Id=2,
                MovieName="Titanic",
                ReviewerName="Rahul Sahay",
                ReviewerComments="Awesome",
                ReviewerRating=5
            },
            new MovieReviews{
```

```
        Id=3,
        MovieName="Die Another Day",
        ReviewerName="Rahul Sahay",
        ReviewerComments="<script>alert('You are hacked!');</script>",
        ReviewerRating=4
    }
};
```

Now, let me build and refresh the page.



As you can see in the above screen shot, that the injected script tag and its value has been automatically sanitized by the razor view engine. So, it took the value, HTML Encode the same and then sent to the browser, hence script gets converted

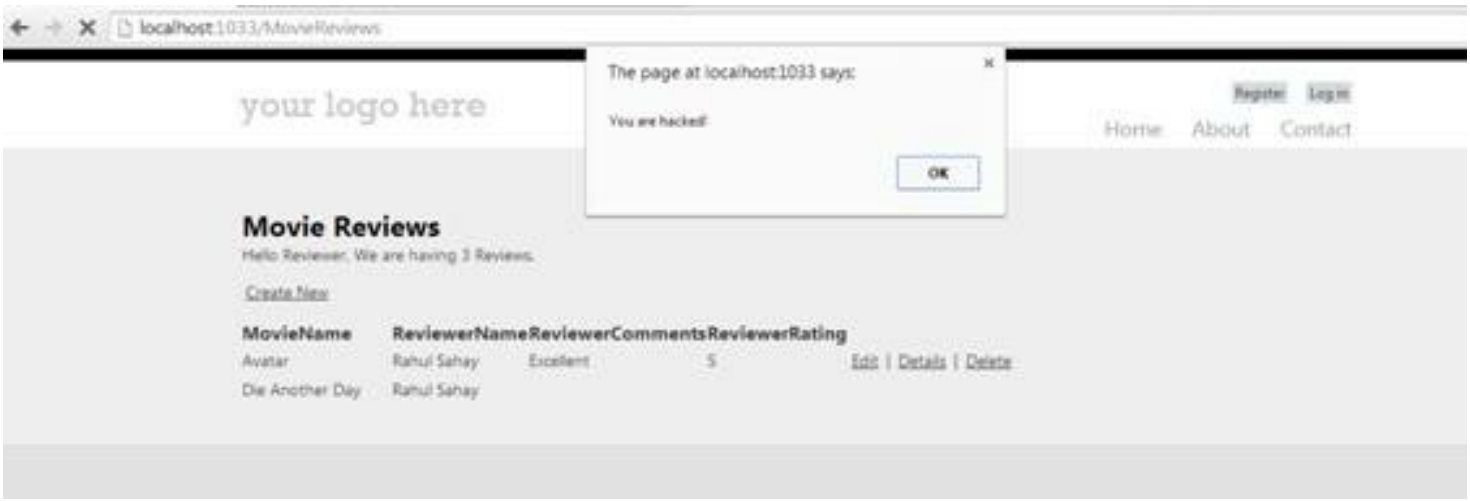
to plain text. So, let's suppose the scenario, if razor would not have taken care this, this screen would have been ended into an annoyance screen as every time you have to encounter with an alert box.

Situation could have been more dangerous if malicious user could somehow reach through your database, issue some malicious command to execute a query and steal the potentially confidential information.

Now, there is also a way which permits the developer to execute script without HTML encoding. So, there is a way to do that by using **@Html.Raw**.

```
<td>
    @*@Html.DisplayFor(modelItem => item.ReviewerComments)*@
    @Html.Raw(item.ReviewerComments)
</td>
```

Now, with the above change in place, when I refresh the page, then it will produce the below output.

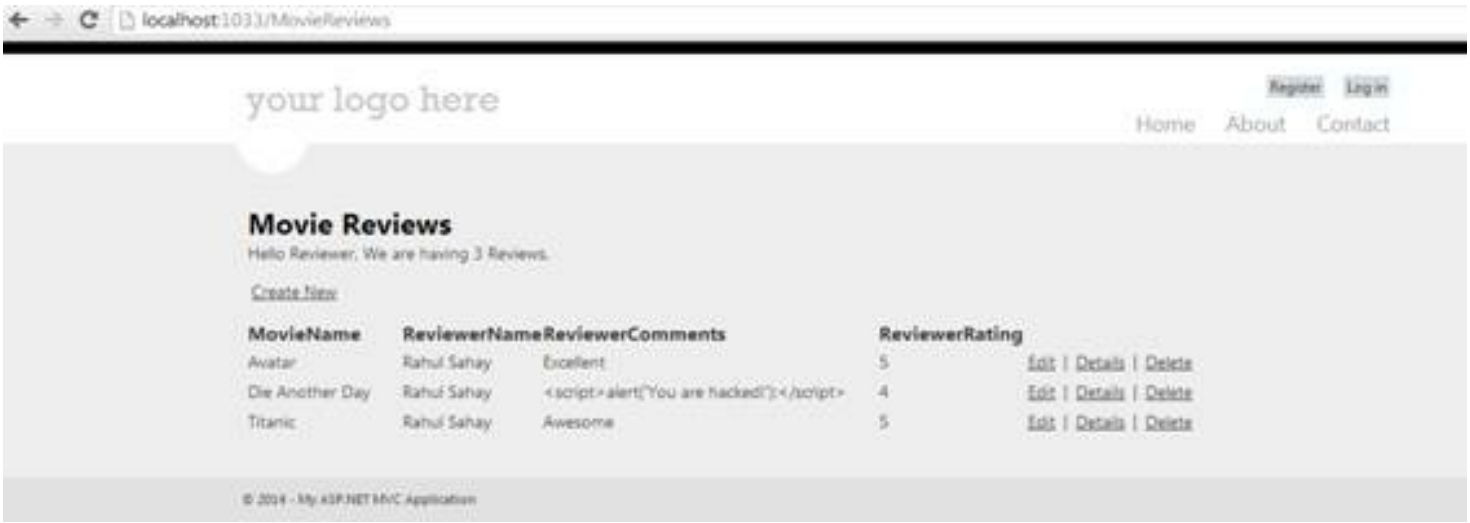


More on Razor Syntax:-

Now, let me show one more thing, besides following the conventional style of outputting the result what Visual

studio has created, we can produce the output implicitly as well. In the below example, it is going to produce me the same reviewer comments, but achieved with minimal coding. Here, Visual Studio is smart enough to identify the item and pass the same to the browser.

```
@item.ReviewerComments
```



Now, let's suppose that you want to mix and match the code expression with

string literals. Let's suppose you need to show the rating out of 5. Then, simply what you can do is modify the expression like this

```
@item.ReviewerRating / 5
```

Now, when I save and refresh the same, it will produce me the below result.



Now, in many cases, this might be possible that you are not dealing rating

kind of scenario; you need to actually perform the mathematical operation and get the output. In that case, in order to achieve the same, we need to mask the same inside the parentheses as shown below also known explicit style.

```
@(item.ReviewerRating / 5)
```

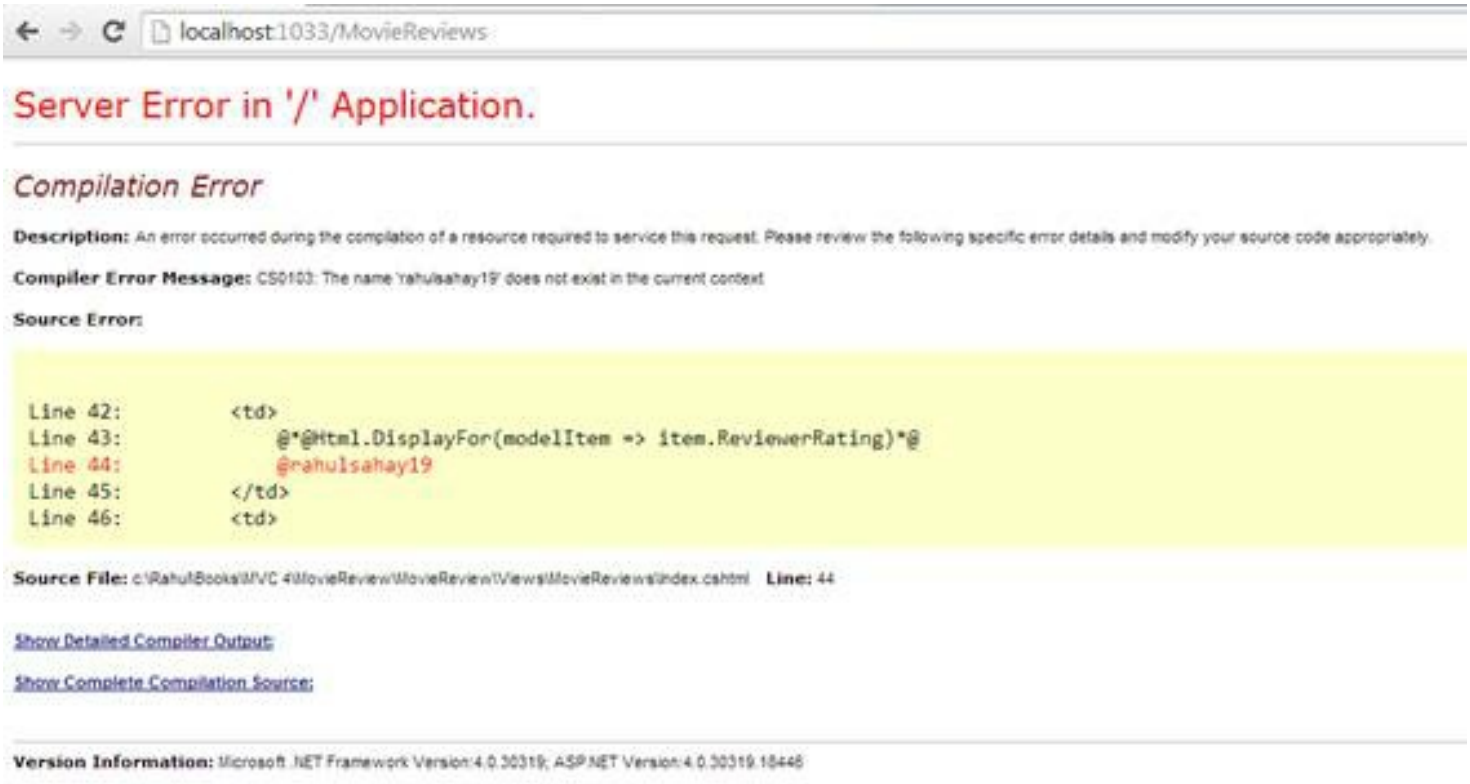
Now, when I save and refresh the same, it will produce me the result of the division.



As you can see in the above screen shot it returned the integer result because we defined rating as **Int** in our model class.

Many a time, wherein you need to provide your contact address as in tweeter handle so that followers can follow you. Default way of mentioning the same is beginning with @ symbol like "***@rahulsahay19***"; my tweeter handle. Now, if I write like these in razor, this will return me an error as shown below.

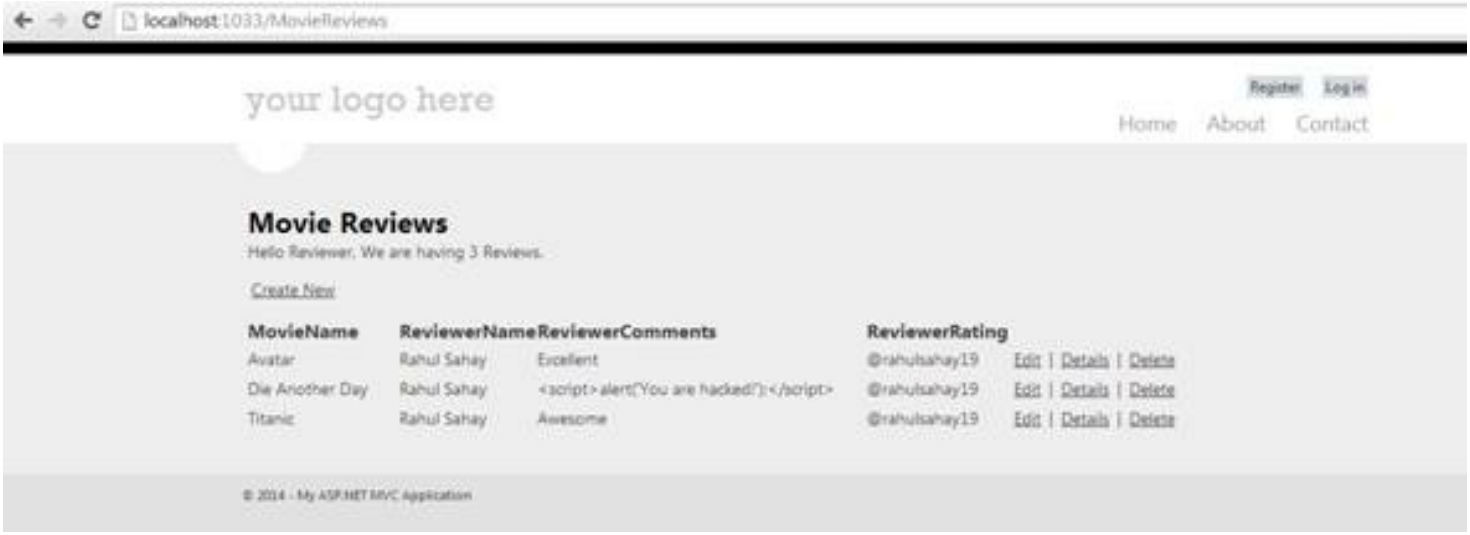
```
@rahulsahay19
```



Now, for this kind of scenario, we need to escaping character again by using the @ symbol twice in this case.

@@rahulsahay19

Now, when I save and refresh the page, it will produce me the desired output.



Like these there could be numerous of code expressions example. Now, let's look another kind of scenarios where in code blocks are useful. Now, let me show you a default and very simple example of code block.

```
@{
    ViewBag.Title = "Movie Reviews";
}
```

A code block always begin with the @

symbol and then the opening curly brace followed by the code expressions and then the closing curly brace just to delimit the code block. Like, these we could have as many code blocks as we want to produce the desired output. However, another mature example of code block could be "***foreach***" statement wherein razor is producing the table with data inside.

```
@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.MovieName)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.ReviewerName)
        </td>
        <td>
            @*@Html.DisplayFor(modelItem => item.ReviewerComments)*@
            @item.ReviewerComments
        </td>
        <td>
            @*@Html.DisplayFor(modelItem => item.ReviewerRating)*@
            @@rahulsahay19
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.Id }) |
            @Html.ActionLink("Details", "Details", new { id=item.Id }) |
            @Html.ActionLink("Delete", "Delete", new { id=item.Id })
        </td>
    </tr>
}
```

Now, as you can see in the above code block, razor is processing the C# code in combination with the string literals and hence producing the output. Hence, it's a fairly good example of mixing and matching the C# code with the string

literals. Now, here you can modify the template entirely and produce the same result inside the div. let me just delete the HTML table and the code block and rewrite everything my custom way.

```
@model IEnumerable<MovieReview.Models.MovieReviews>

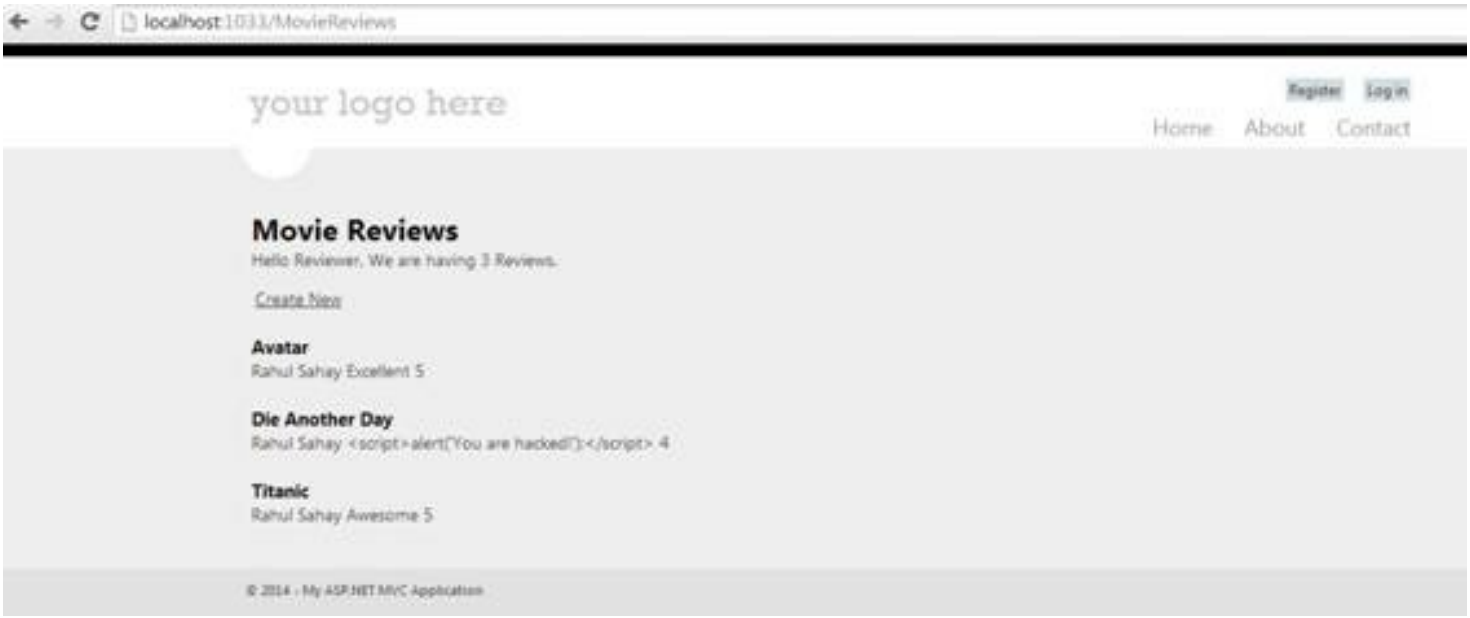
@{
    ViewBag.Title = "Movie Reviews";
}

<h2>Movie Reviews</h2>

Hello Reviewer, We are having @Model.Count() Reviews.
```

```
<p>
    @Html.ActionLink("Create New", "Create")
</p>
@foreach (var item in Model)
{
    <div>
        <h4>@item.MovieName</h4>
        <span>@item.ReviewerName</span>
        <span>@item.ReviewerComments</span>
        <span>@item.ReviewerRating</span>
    </div>
}
```

Now, here in the above snippet, I have replaced the HTML table with small code block outputting the same but in a different context. So, now, when I save the same and refresh, it will produce the desired result.



Now, sometimes what happen while evaluating the expression from C# to string literal back and forth, razor gets confused with the intension of user like shown below

```
<p>
    @Html.ActionLink("Create New", "Create")
</p>
@foreach (var item in Model)
{
    <div>
        <h4>@item.MovieName</h4>
        <span>@item.ReviewerName</span>
        <span>@item.ReviewerComments</span>
        <span>@item.ReviewerRating</span>
    </div>
}
```

Now, here my intention is to produce the word "***Review***" with each div as a separator, but when I save the same and refresh, this will again produce me the below parsing error.



Now, in order to fix the same, I need to again use of some sort of escaping character to tell the razor to interpret the same as string literal rather than C# code. So, to fix the same I need to use "**@:**" symbol before the string literal as

```
@:Review
```

Now, when I save it and refresh the

same, it will produce me the desired result.



Although, presentation is not pleasing at the moment, but we'll fix the presentation in the coming chapters when we'll be discussing about bootstrapping the UI. But, for now just focus on Razor Views and its characteristics.

Layout Views:-

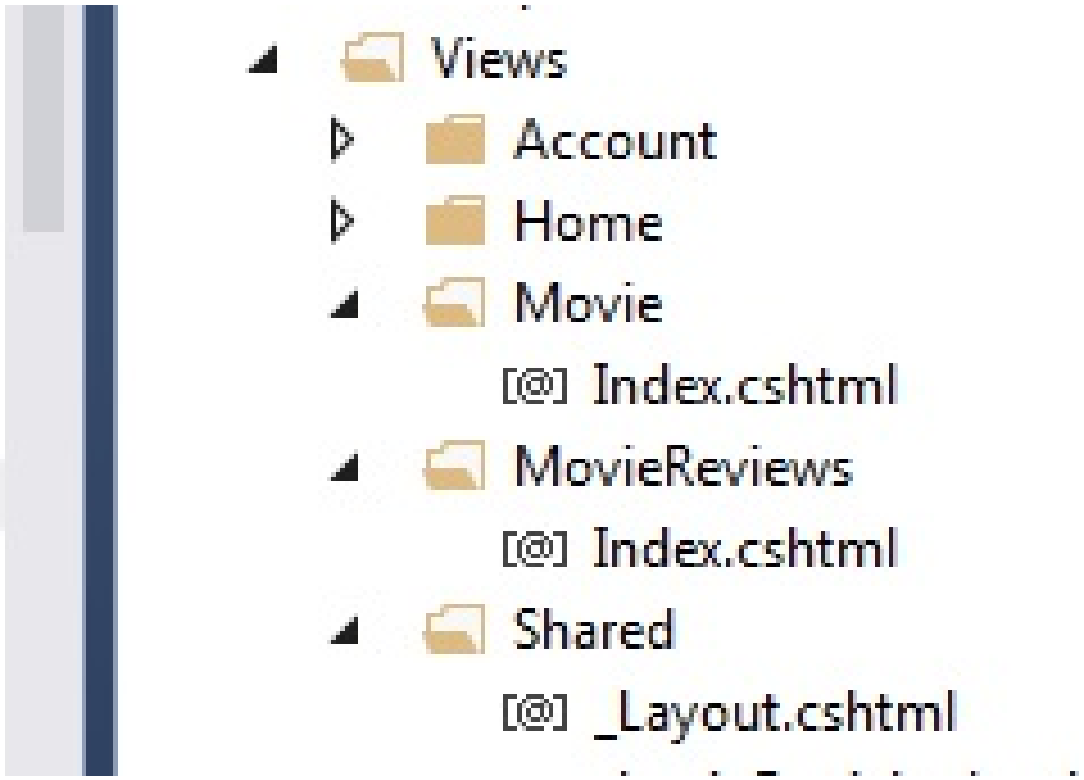
Now, let's focus on Layout views, Layouts are nothing but master pages. If you have been an ASP.Net Developer, then you must be familiar with **Master Pages**. However, Master Pages are the ones which decide the overall look and feel of the site. In the similar context, we have Layout views in ASP.Net MVC. Layout views also decides the overall look and feel, navigational structure of the site, Headers, Footers and many other things which could be placed globally are placed in Layout page.

Now, in the layout page there are two important sections. These are

RenderBody

RenderSection

So, basically above listed sections allow the content views like the Index view to get injected at the specific section of the page. You can find the layout view in shared repository as shown below.



Now, when you open this page, you will see the complete sections rendering different pieces at different locations of the page.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <title>@ViewBag.Title - My ASP.NET MVC Application</title>
        <link href=" /favicon.ico" rel="shortcut icon" type="image/x-icon" />
        <meta name="viewport" content="width=device-width" />
        @Styles.Render("~/Content/css")
        @Scripts.Render("~/bundles/modernizr")
```

```
    </head>
    <body>
        <header>
            <div class="content-wrapper">
                <div class="float-left">
                    <p   class="site-title">@Html.ActionLink("your   logo   here",
"Index", "Home")</p>
                </div>
                <div class="float-right">
                    <section id="login">
                        @Html.Partial("_LoginPartial")
                    </section>
                    <nav>
                        <ul id="menu">
                            <li>@Html.ActionLink("Home", "Index", "Home")</li>
                            <li>@Html.ActionLink("About", "About", "Home")</li>
                            <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
                        </ul>
                    </nav>
                </div>
            </div>
        </header>
        <div id="body">
            @RenderSection("featured", required: false)
            <section class="content-wrapper main-content clear-fix">
                @RenderBody()
            </section>
        </div>
        <footer>
            <div class="content-wrapper">
                <div class="float-left">
                    <p>&copy; @DateTime.Now.Year - My ASP.NET MVC Application</p>
                </div>
            </div>
        </footer>

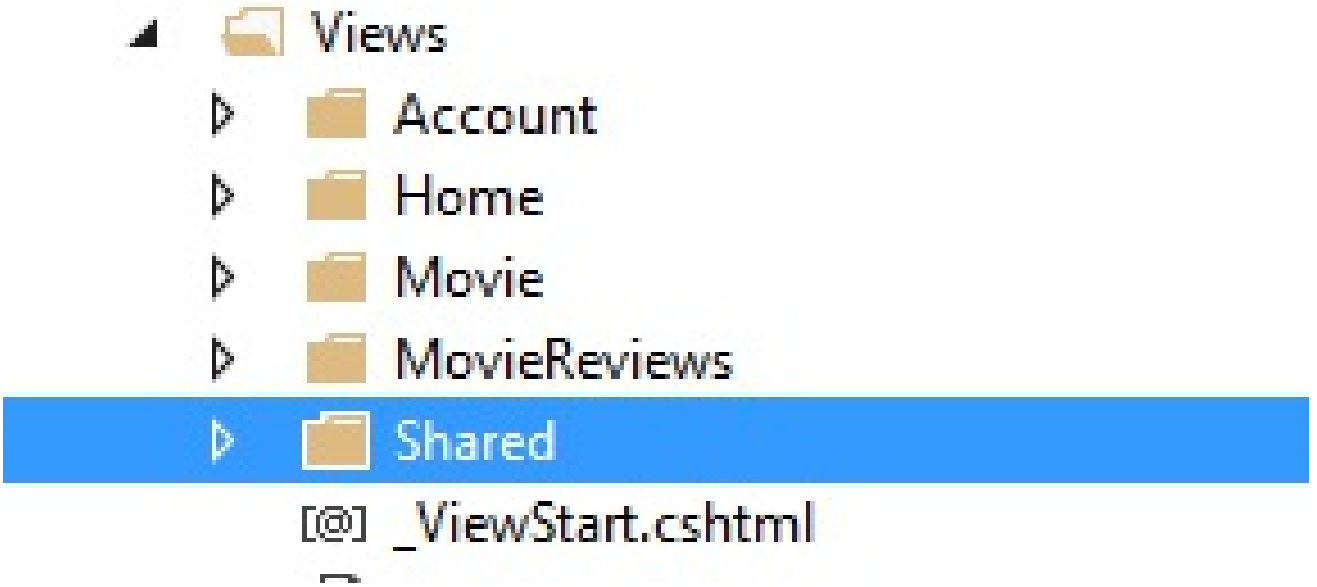
        @Scripts.Render("~/bundles/jquery")
        @RenderSection("scripts", required: false)
    </body>
</html>
```

In a nutshell, if I have to explain the

Layout view, there is nothing special about this razor view, it has literal texts, it has C# code expressions, it can have code blocks as well. It's the file where you will see **Doctype**, **Head Tag**, and then the definition for the body of the element. And please be informed that if you make any changes to the layout view that will reflect application wide.

Currently, in this application we are having only layout page, but we can have multiple layout pages as well. In the above code snippet of layout view, you will see one "***@RenderBody()***" element, this **@RenderBody** is responsible for injecting content views like Index view.

Now, you might be thinking that how ASP.Net MVC Framework knows that how to pick up this layout view, as there is no instruction written neither in the Layout view nor in any index view. And to answer this question, let me show you one magical file "***_ViewStart.cshtml***" that sits in the root of the Views folder as shown below.



Now, when you open this file, you will find the declaration to use the above

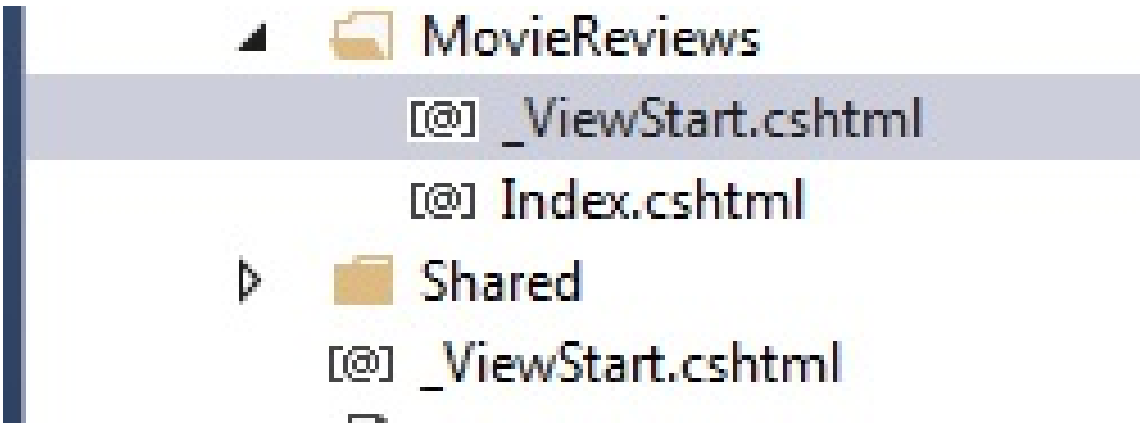
layout file.

```
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```

So, the formula behind this is, that "***_ViewStart.cshtml***" executes 1st than any other view gets start rendering. Hence, this file sets the layout property. So, this is the convention of ASP.Net MVC Framework, whatever you put in the code block of "***_ViewStart.cshtml***" , will execute 1st.

Now, if I have to create multiple layout views across our application, I can follow the below steps.

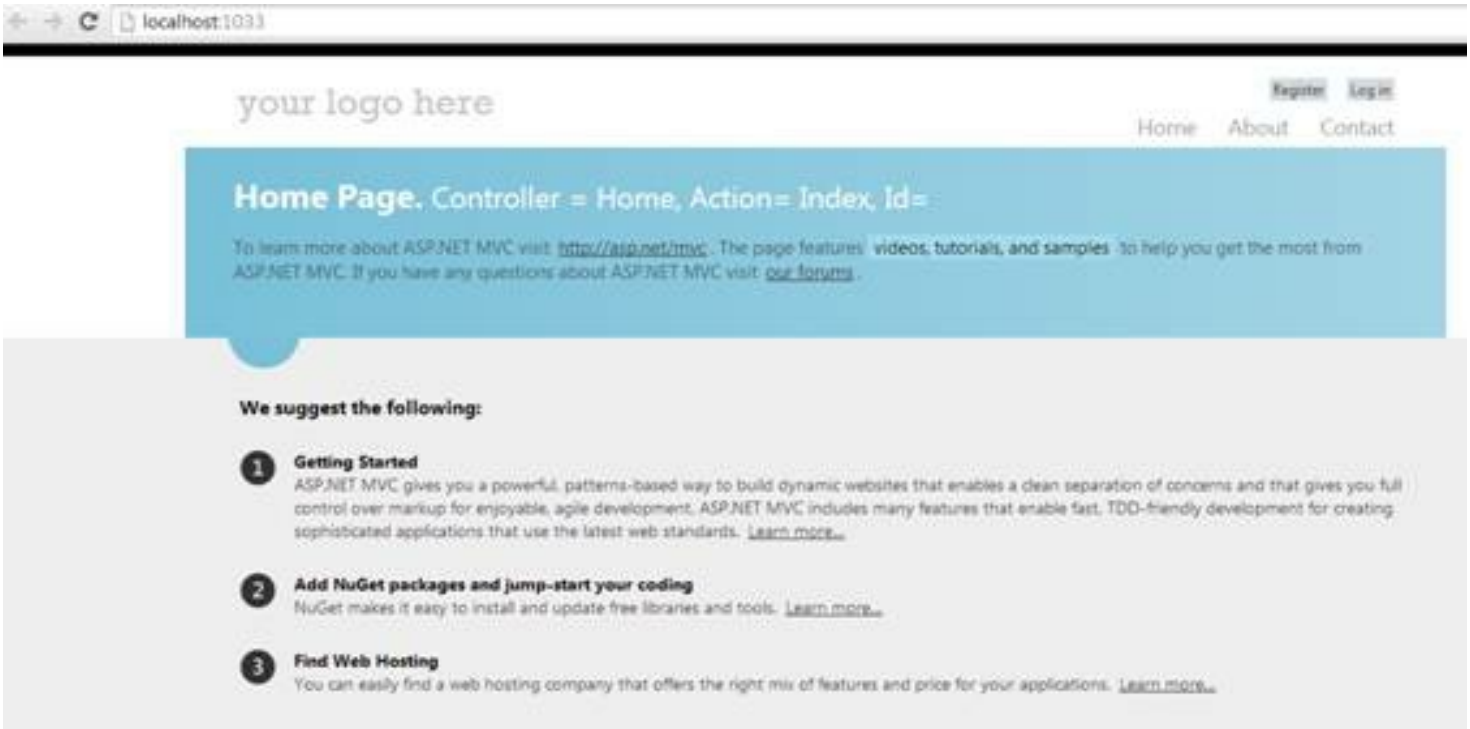
1st Copy the **_ViewStart.cshtml** and then place in the sub directory, where in you want have a different layout. In our case, I have chosen MovieReviews.



Now, open this _ViewStart.cshtml file and set the Layout property by giving the layout filename a suitable name like shown below.

```
@{
    Layout = "~/Views/Shared/_Layout_Movie.cshtml";
}
```

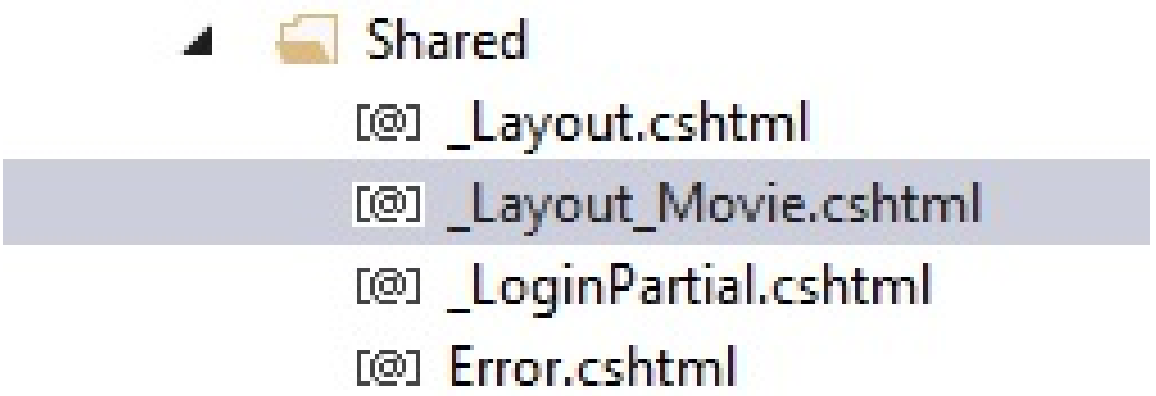
Now, when I save everything and run the app, it will produce me the below output when I render the home page.



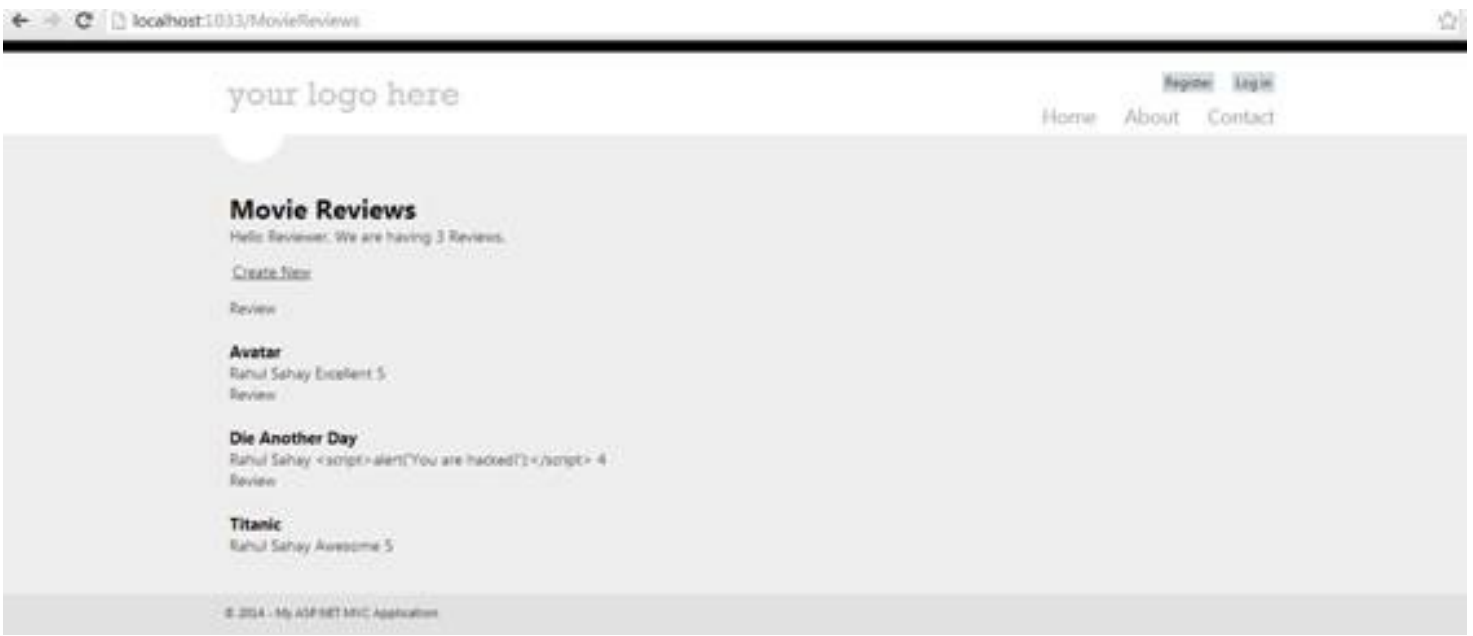
And , when I go to the MovieReviews link, it will present me the below error



And the reason for this is pretty obvious, because here MVC Framework is looking for the new layout, page, which I have not created and hence it resulted me the error. Now, in order to fix this, let me just copy and paste the same layout page but save this with above mentioned name.



Now, when I run the app, it will present me the same User Interface, but now with a different Layout page.



So, with this way we can go ahead and make the multiple layout pages in our

application. Apart from this, I can also change the layout view per view basis, means I can tell any custom view; which layout view to render. Let me show you one,

```
@model IEnumerable<MovieReview.Models.MovieReviews>

@{
    ViewBag.Title = "Movie Reviews";
    Layout = "~/Views/Shared/_Layout_another.cshtml";
}

<h2>Movie Reviews</h2>

Hello Reviewer, We are having @Model.Count() Reviews.
<p>
    @Html.ActionLink("Create New", "Create")
</p>

@foreach (var item in Model)
{
    @:Review
    <div>
        <h4>@item.MovieName</h4>
        <span>@item.ReviewerName</span>
        <span>@item.ReviewerComments</span>
        <span>@item.ReviewerRating</span>
    </div>

}
```

Now, here in the above example I have

literally override the Layout property. Now, this will look for layout view with the name _Layout_another, which doesn't exist in the system, hence this will return an error as shown below.



I can also turn off the layout view as shown below.

```
@model IEnumerable<MovieReview.Models.MovieReviews>

@{
    ViewBag.Title = "Movie Reviews";
    Layout = null;
}

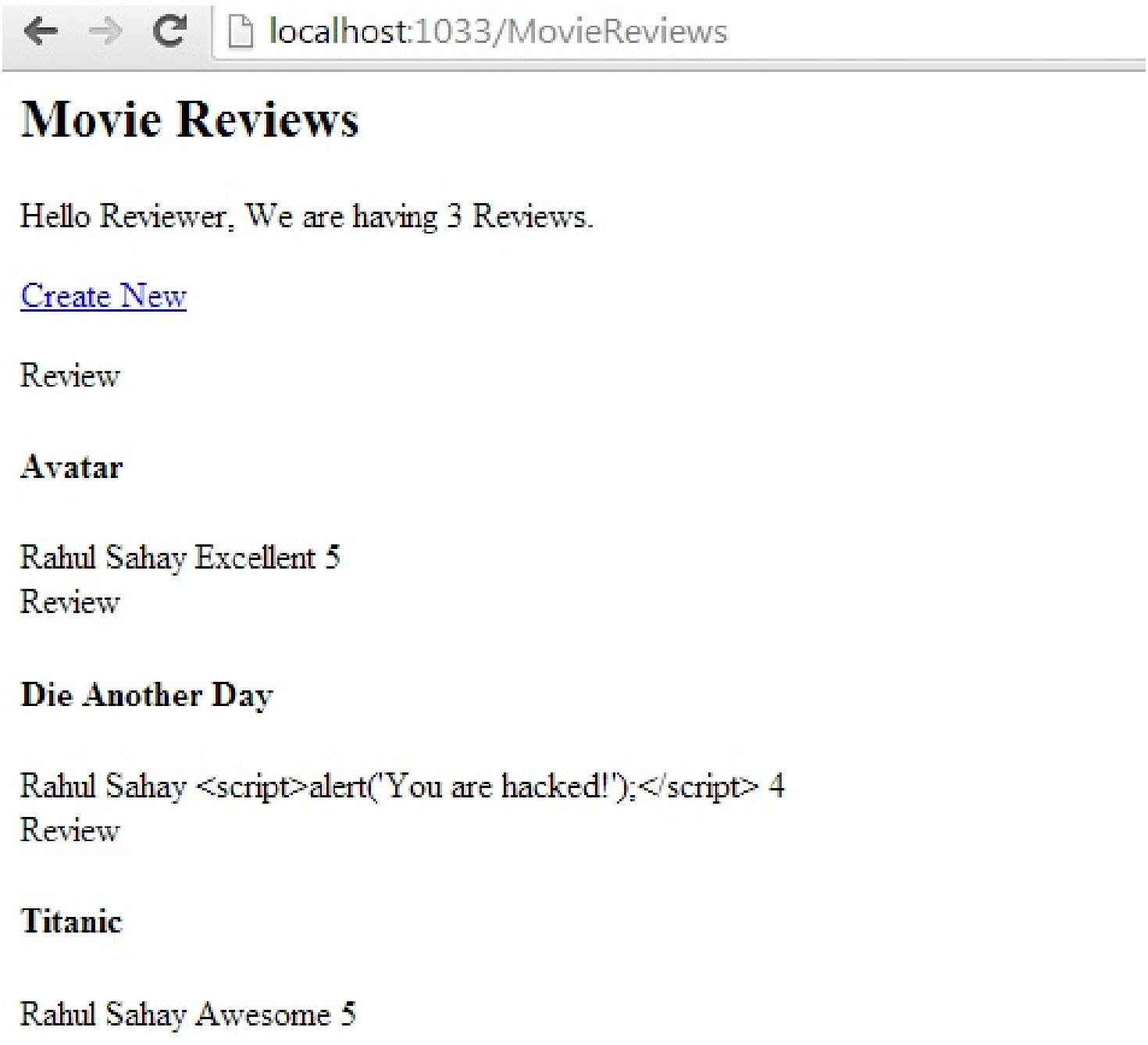
<h2>Movie Reviews</h2>

Hello Reviewer, We are having @Model.Count() Reviews.
<p>
    @Html.ActionLink("Create New", "Create")
</p>

@foreach (var item in Model)
{
    @:Review
    <div>
        <h4>@item.MovieName</h4>
        <span>@item.ReviewerName</span>
        <span>@item.ReviewerComments</span>
        <span>@item.ReviewerRating</span>
    </div>
}
```

Now, when I save and refresh the page,

it will produce me the below output.



No styling, no theme, no navigational links, just plain html. Now, inside the

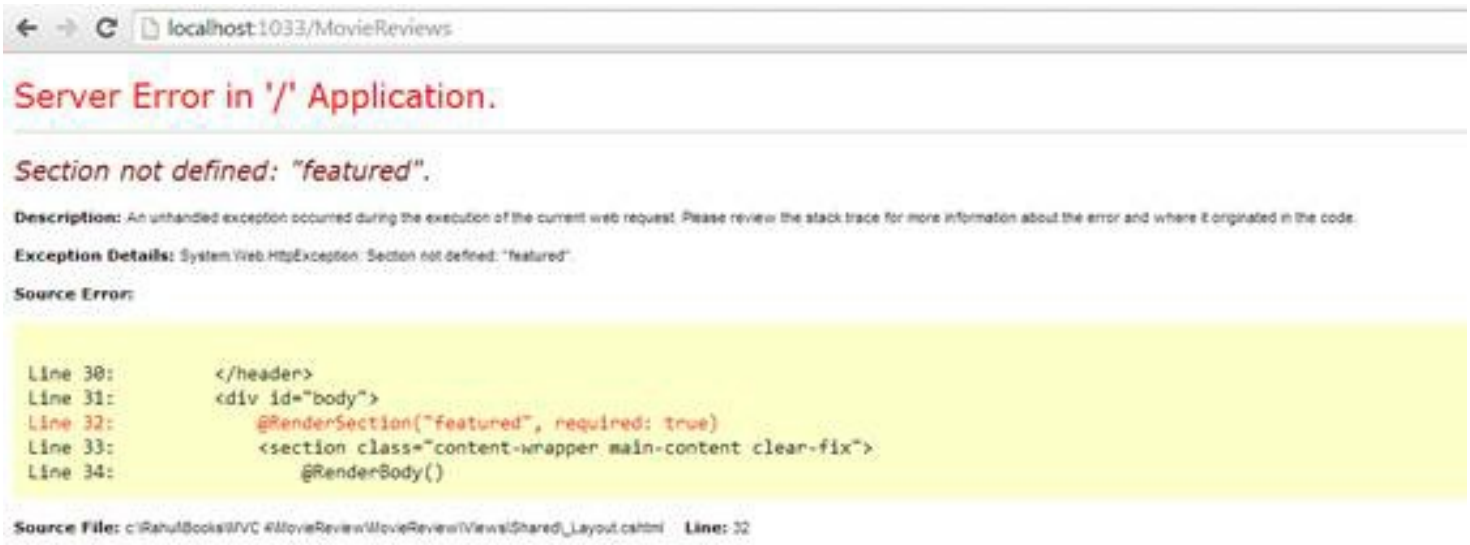
layout view, there is one more method called as "***@RenderSection***". One point to understand here, that this method is optional; you can have multiple sections on a page. So, when you see the Layout page, you notice two "***RenderSections***", rendering two different sections at two different places.

1st one is rendering `@RenderSection("featured", required: false)` and 2nd one at the bottom of the page which is rendering scripts as `@RenderSection("scripts", required: false)`.

Now, if you see the code closely, you code easily observer that currently **RenderSection** is marked as optional, but, what will happen if you switch this to true. Let's try and check the same.

```
@RenderSection("featured", required: true)
```

Now, when I save the page and try, then it will produce me the below error.



Now, this error is quite educational, it's clearly saying that you have marked Section field as mandatory in the layout page, but the same has not created on the index page. So, in order to fix the same I can go ahead and create one section there on the reviews page.

```
@model IEnumerable<MovieReview.Models.MovieReviews>

@{
    ViewBag.Title = "Movie Reviews";

}

<h2>Movie Reviews</h2>
@section featured{
    Hello Reviewer, We are having @Model.Count() Reviews.
}
<p>
    @Html.ActionLink("Create New", "Create")
</p>

@foreach (var item in Model)
{
    @:Review
    <div>
        <h4>@item.MovieName</h4>
        <span>@item.ReviewerName</span>
        <span>@item.ReviewerComments</span>
        <span>@item.ReviewerRating</span>
    </div>
}
```

Now, with the above change in place, when I refresh the page, it will produce

me the desired output.

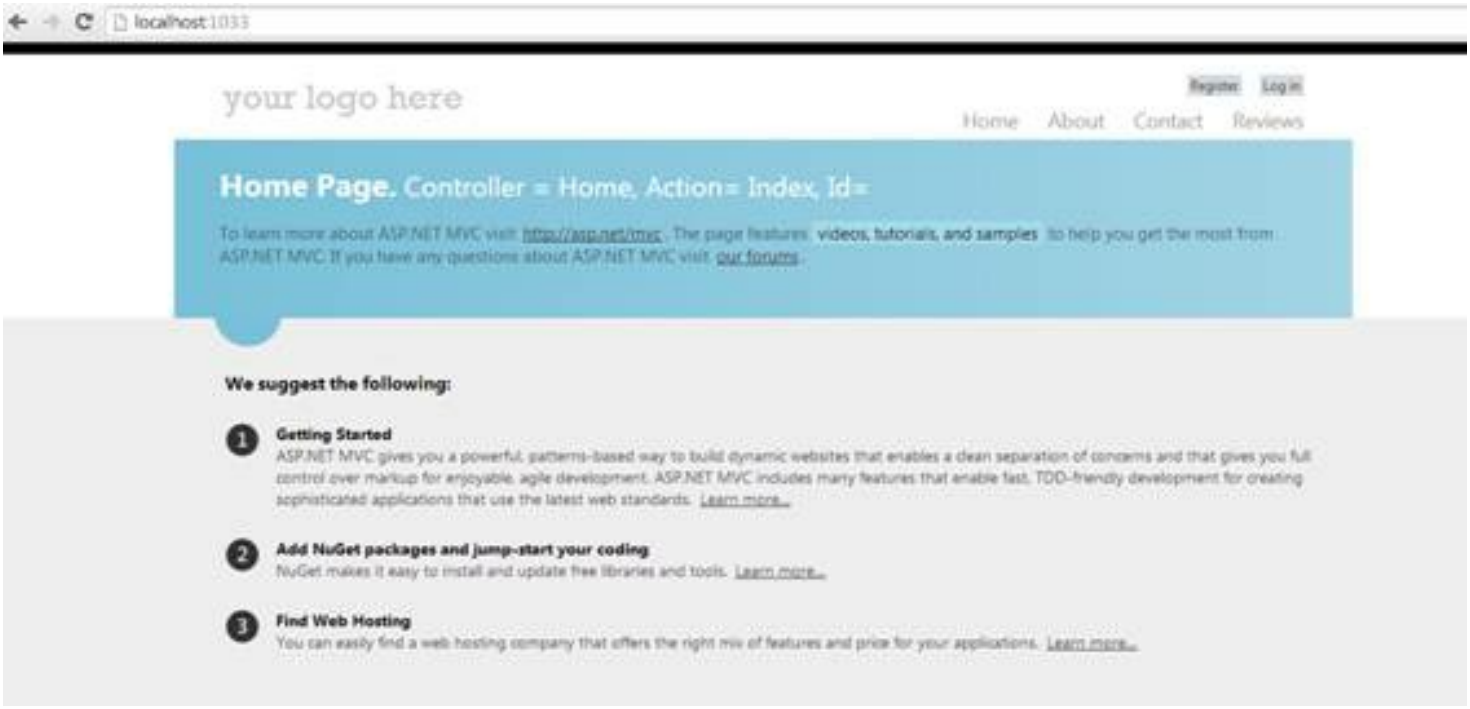


Obviously, here UI got distorted, but for now UI is not my concern. Now, since I have marked the section mandatory on the layout page, hence this error will appear on every page wherever the section has not been implemented, hence it is advisable to turn off this from the layout page. Now, the cool thing about the sections are that these can be placed

anywhere on the page and you can have any number of sections on the page. Now, let me go and create one link for our MovieReviews page. I just need to make one small change.

```
<nav>
                    <ul id="menu">
                        <li>@Html.ActionLink("Home", "Index", "Home")</li>
                        <li>@Html.ActionLink("About", "About", "Home")</li>
                        <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
                        <li>@Html.ActionLink("Reviews",                "Index",
"MovieReviews")</li>
                    </ul>
                </nav>
```

Now, when I save the same and refresh the page, it will create a new link for my page as shown below.



HTML Helpers:-

As you might have noticed in the above explanation, where in I have used Action Link to create one navigation property is one of the many HTML Helpers. The purpose of the HTML Helpers is pretty simple; it makes it easy to create small chunks of HTML. In order to perform the

same, we have variety of helpers. Some of them I have listed below:

Createinputs

Createlinks

Createforms

Also, there are helpers which are meant for validation messages, labels etc. some of these helpers are intelligent enough to resolve the dependency quite brilliantly like shown below

(@HTML.EditorFor(model=>model.M

So, if you see the above Editor For property it's job is quite simple, it will walk up to the property and inspect it's type; determined that "***MovieName***"

property is a string and hence will emit "***Textbox***" for editing the same. Consider, the same case, if it would have been with some Boolean type, and then it would have emitted the "***Checkbox***". There are tons of other properties which **EditorFor** performs, but it would be quite significant to explore the same when we use it later in coming chapters while developing the application.

Now, in a nutshell **EditorFor**, **LabelFor**, **ValidationSummary**, all these are available from the HTML properties which View inherits. Now, let's go ahead and use the same in our application, so that you can understand

the concepts quite wisely.

So, what I am going to do here, I am going to enable the user to edit the reviews. So, this edit action will be using significant numbers of different HTML Helpers.

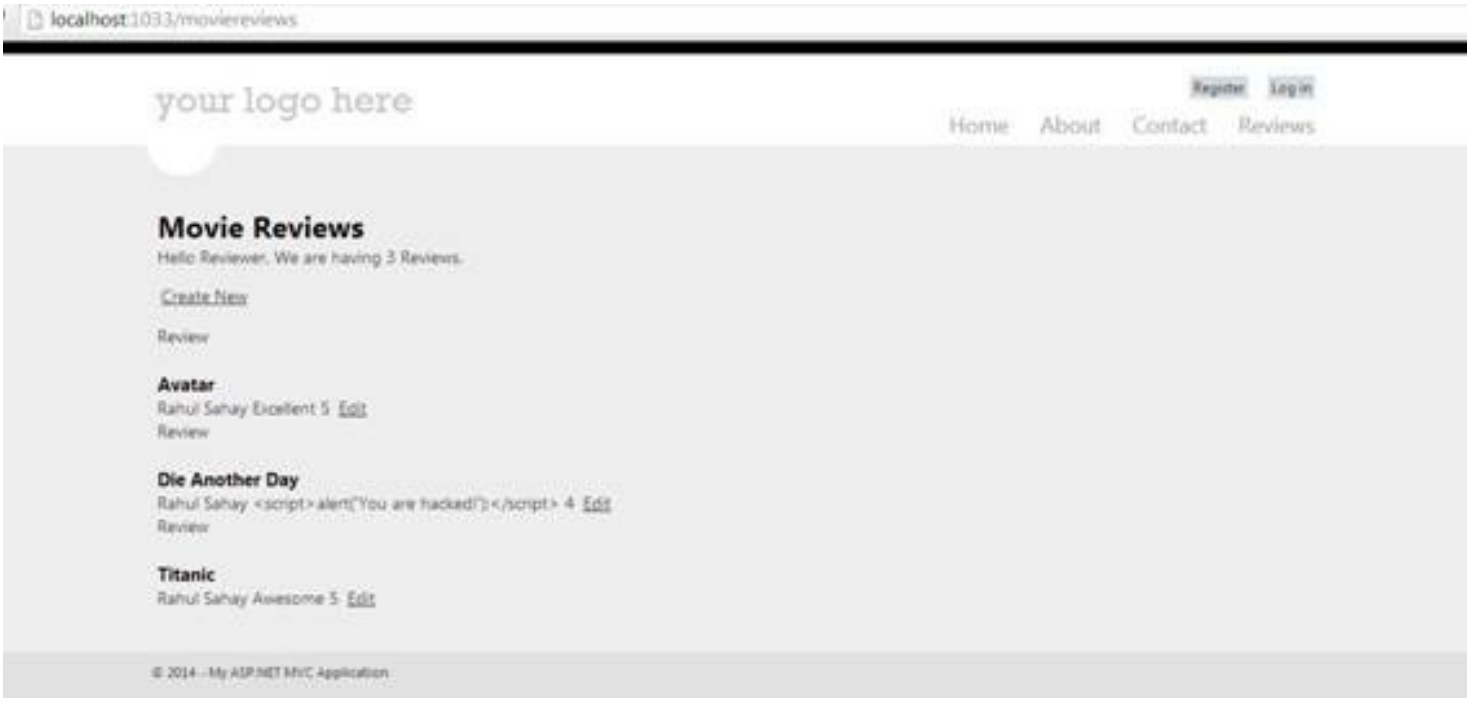
```
<span>@Html.ActionLink("Edit","Edit",new{Id=item.Id})</span>
```

As you can see in the above snippet, I have created one more span tag for editing the review. So, with the action link property obviously 1st property is going to be the **name** of the link, as "***Edit***", 2nd property is going to be the **action method** in the **Controller** which is again going to be "***Edit***", 3rd

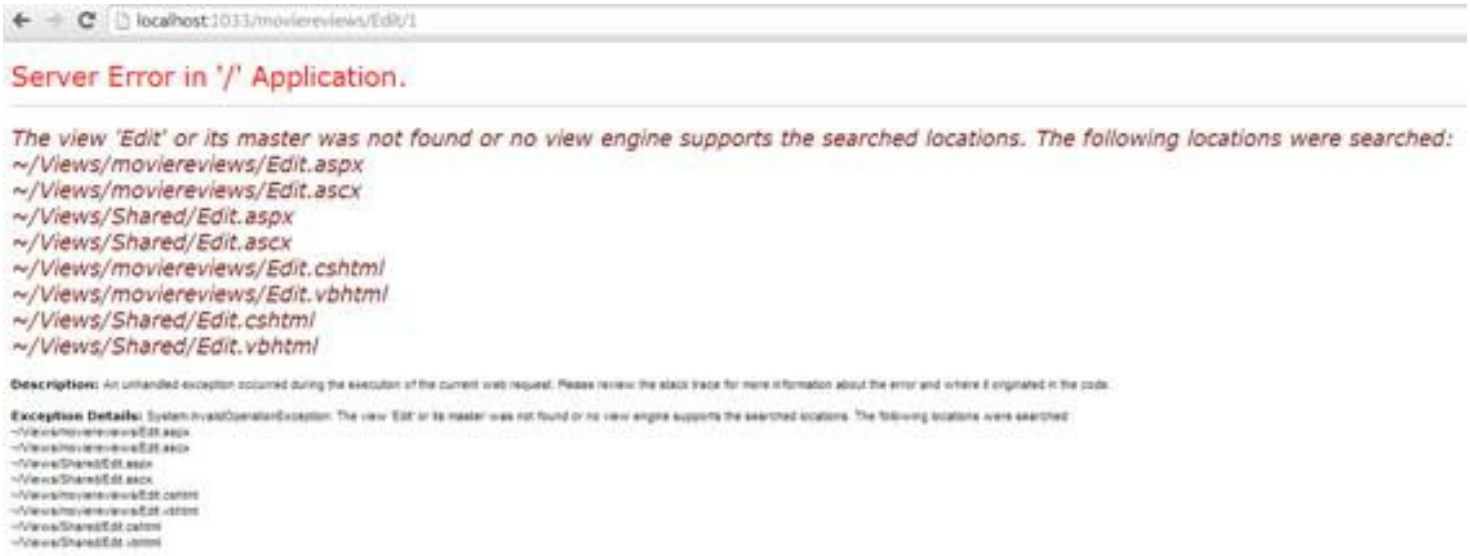
parameter is the one where I need to pass along few values with this action link, so that MVC framework can identify which review am interested in to modify, hence I have passed **Id** as a value.

Now, as you might have noticed that I have encapsulated the **Id** property in an anonymous object. And there can be any no of different values which we want to pass, we can send the same way. So, ultimately these additional information will get passed to the routing engine, and then routing engine will decide that ok these additional values need to get passed in the URL. So, the pattern of URL would be like

"***moviereviews/edit/1***". Now, let's save everything and refresh the page.



Now, when you hover on any edit link, this will display the above highlighted link at the bottom of the page. Now, let me go ahead and click the same. But, when I click it, it will present me the below error.



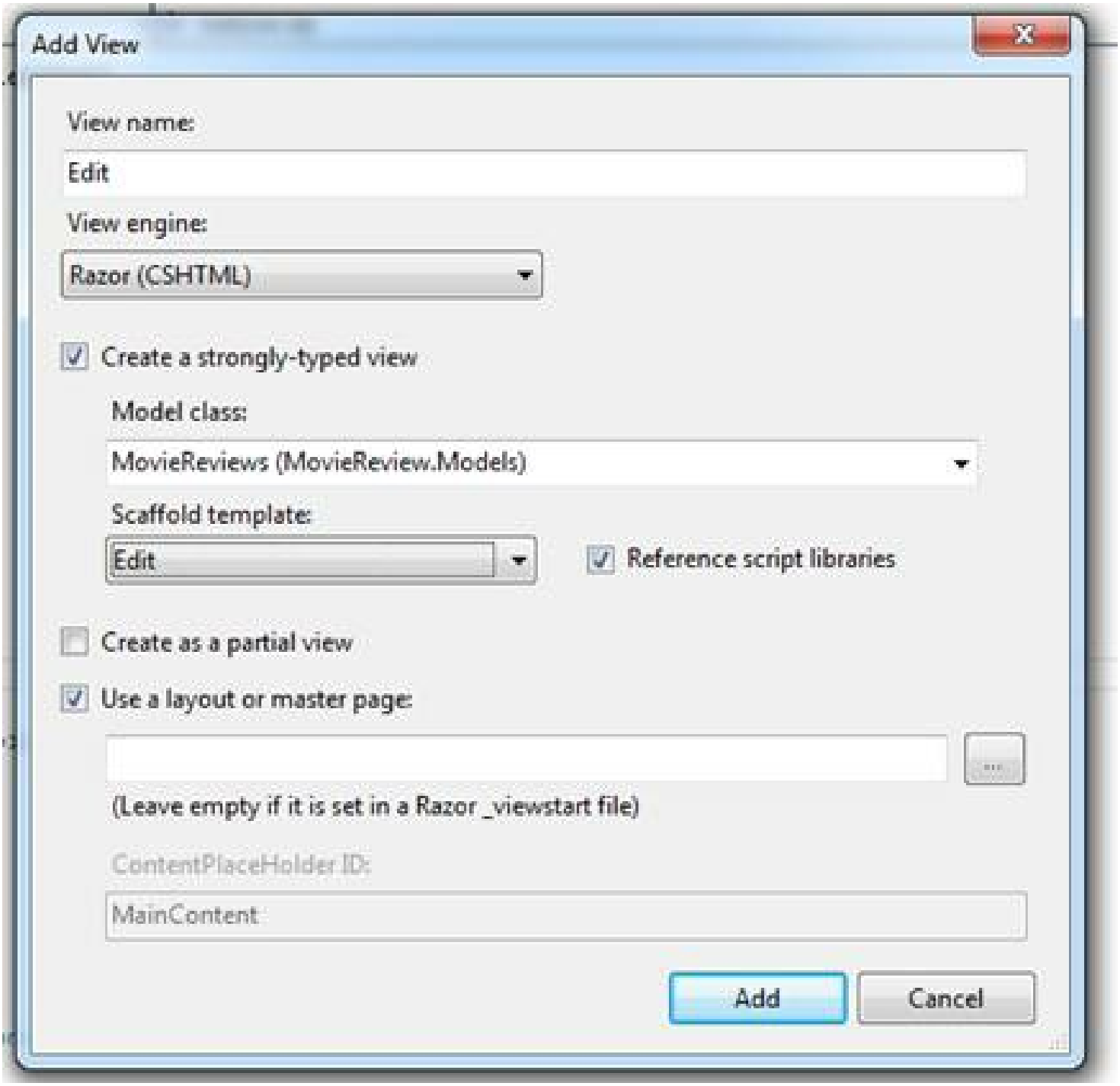
And now the reason for the above error is pretty obvious, we already have the Edit action in our controller, but we don't have the view associated with this. So, in order to fix it, let me go ahead and create the view 1st. obviously the technique to create the view is going to be same. But, before this we need to add few more lines of code in the edit action. So, the 1st thing where I'll focus is action method for the "***Get***" action. Now, the **Id** in the **URL** will automatically get

extracted from the parameter which is listed in the below method. Hence, what I need to do is to go ahead and find the id which is invoked.

```
//
      // GET: /MovieReviews/Edit/5

      public ActionResult Edit(int id)
      {
          var movieReview = _movieReviews.Single(m => m.Id == id);
          return View(movieReview);
      }
```

So, above code will return me a single review associated with the **id**. Now, the above snippet is just a little **LINQ** query to fetch the review against the **id** which is getting received. Now, this fetched result is going to be the model for my view. Now, I can go ahead and add **Edit** view as shown below.



Now, this will create me the desired view as shown below.

```
@model MovieReview.Models.MovieReviews

@{
    ViewBag.Title = "Edit";
}

<h2>Edit</h2>

@using (Html.BeginForm()) {
    @Html.AntiForgeryToken()
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>MovieReviews</legend>

        @Html.HiddenFor(model => model.Id)

        <div class="editor-label">
```

```
            @Html.LabelFor(model => model.MovieName)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.MovieName)
            @Html.ValidationMessageFor(model => model.MovieName)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.ReviewerName)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.ReviewerName)
            @Html.ValidationMessageFor(model => model.ReviewerName)
        </div>

        <div class="editor-label">
            @Html.LabelFor(model => model.ReviewerComments)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.ReviewerComments)
            @Html.ValidationMessageFor(model => model.ReviewerComments)
        </div>

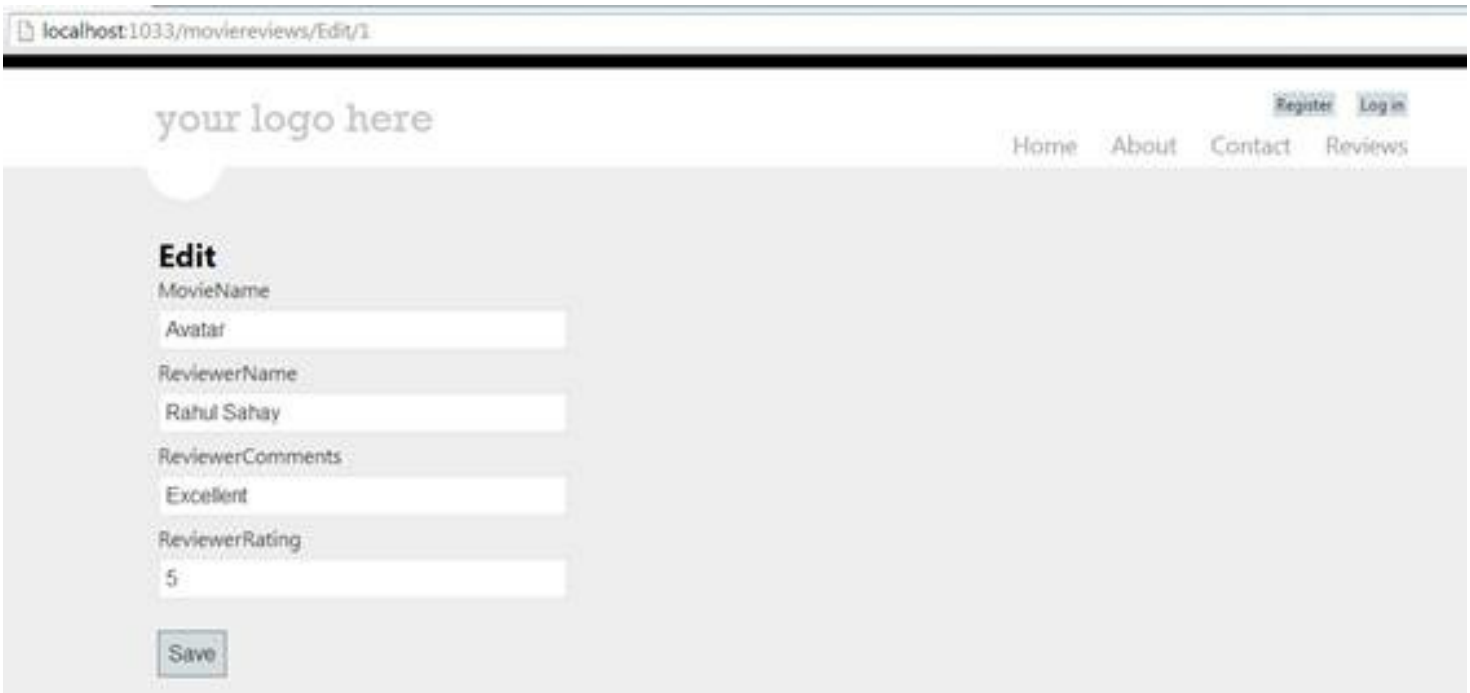
        <div class="editor-label">
            @Html.LabelFor(model => model.ReviewerRating)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.ReviewerRating)
            @Html.ValidationMessageFor(model => model.ReviewerRating)
        </div>

        <p>
            <input type="submit" value="Save" />
        </p>
    </fieldset>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

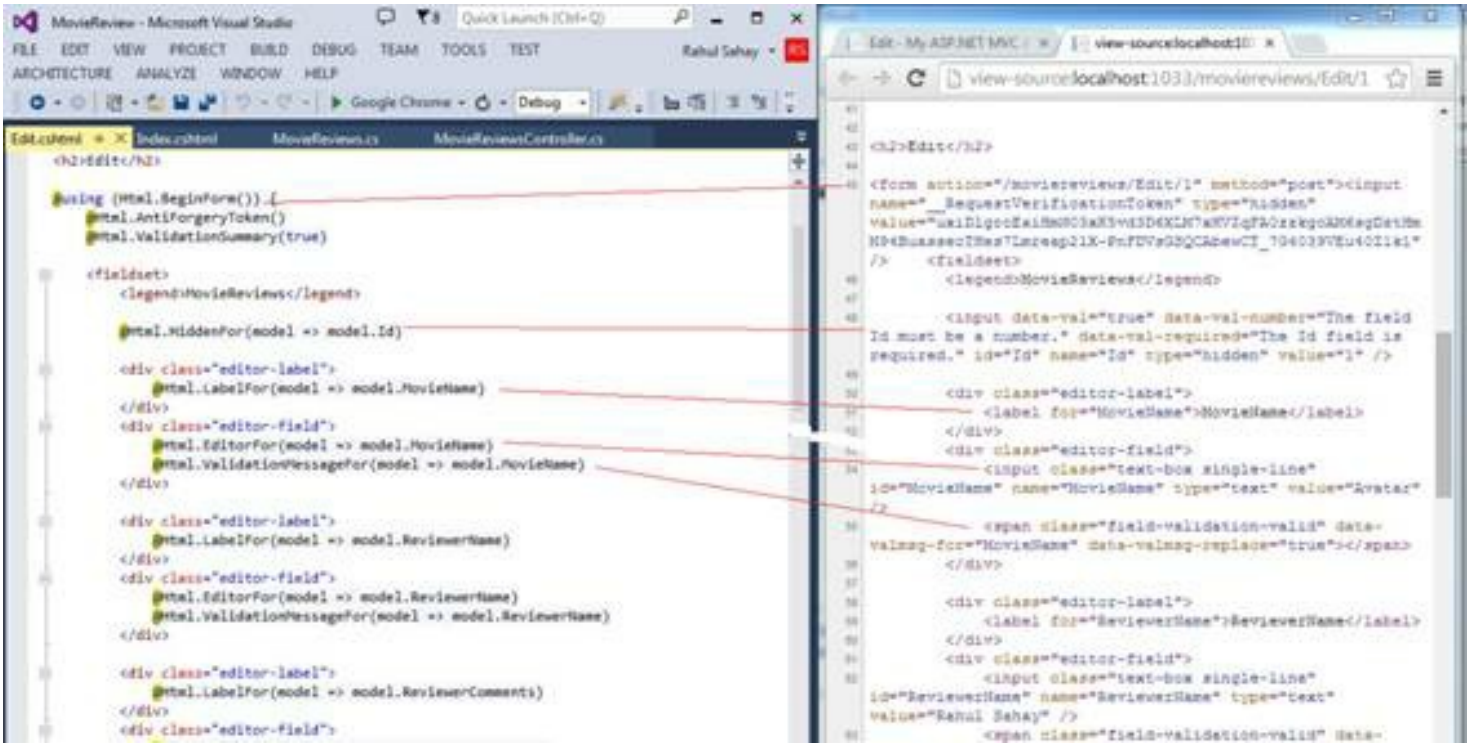
@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Now, when I build and refresh the page, then in that case it will produce me the below output.



So, if you remember above I have explained about "***EditorFor***". Now, based on the incoming value type like **STRING** type for "***MovieName***" property or "***ReviewerRating***" which is of type INT, MVC Framework emitted

me the textbox to edit the same. So, this is what I emphasized about intelligent helpers. Now, let me go ahead and inspect the created form inside the view side by side with the view source.



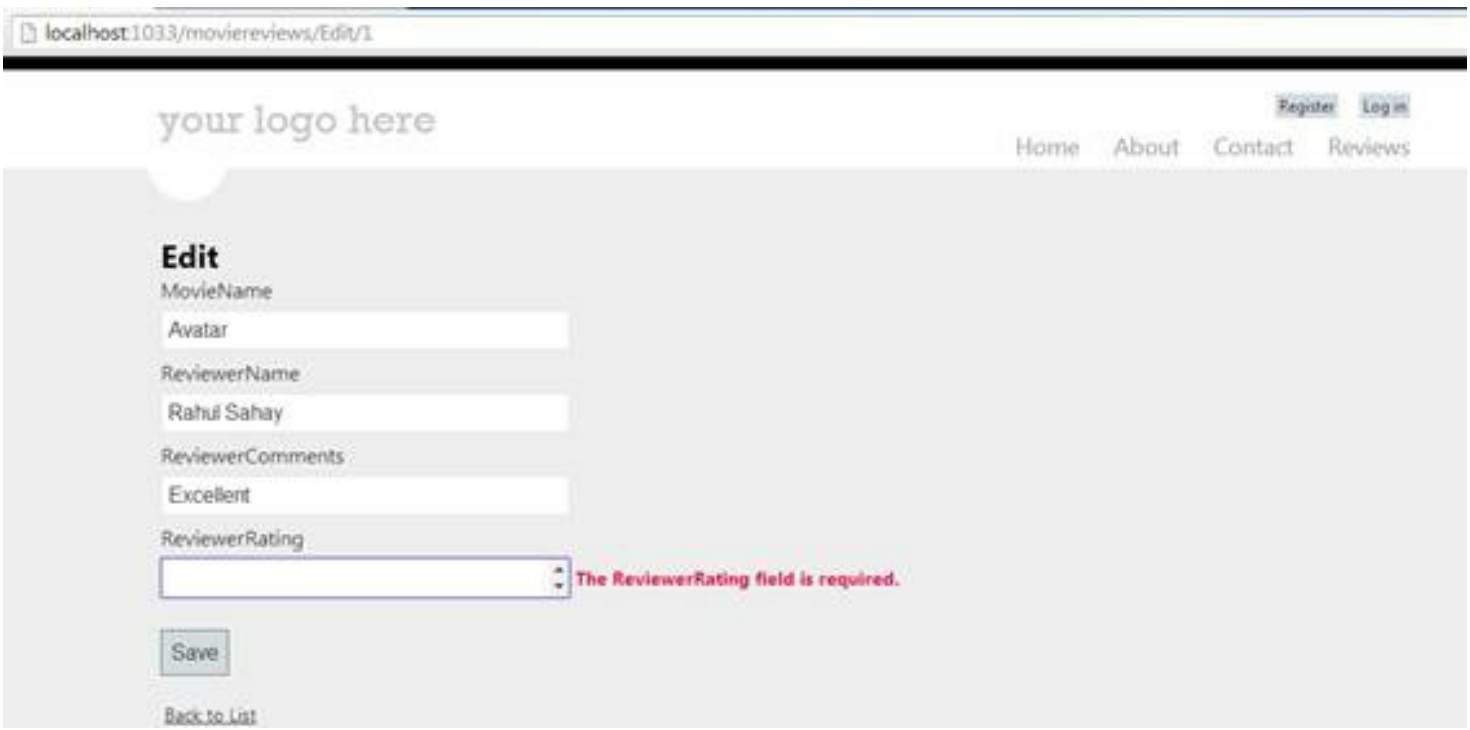
So, here "***HTML.BeginForm***" writes an opening **Form Tag** for me. It also ensures that method is going to be "***Post***", when user clicks on the "***Save***"

button. One point to note here that "***BeginForm***" is very useful, because it enable me to change the URL, change the method etc. Also, there is one more HTML helper listed in the view that is "***@HTML.HideenFor***" which is going to emit the hidden field. Why MVC Framework has emitted a hidden value for this?

MVC Framework intelligently figures out that this is **Id** field mentioned in the model class and hence it needs to be prevented from getting edited, that is why marked as hidden. Then, we have "***LabelFor***" HTML Helper which is emitting the labels on the screen. There is also "**ValidationMessageFor**" helper

which is going to display proper error message if anything wrongly entered while making the change.

Let's take an example, I have omitted the rating value and it intelligently produced me the below output.



Now, this is client side validation which is getting generated from the client side

scripts bundled under "**jqueryval**" as shown below in the script.

```
@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Now, let's go ahead and complete other pending actions like when you post the edited form then it needs to take the effect. So, I am going to modify the default code presented to me by Visual Studio. Here what I am going to do; take the value that user typed in based on the "***ID***" and then move the values in using the method "***TryUpdateModel***". So, what this method will do is go through the process of "***Model Binding***". Now, one point to note that model binding can

happen any time if you have any parameter also in the action method. Also, this is what ASP.Net MVC framework does, it goes there and looks for the request and tries to find the thing and move the object for you. So, in the below method, I have parameter with the name "***ID***" in the model action, ASP.Net MVC will find that ID, and move the same for me.

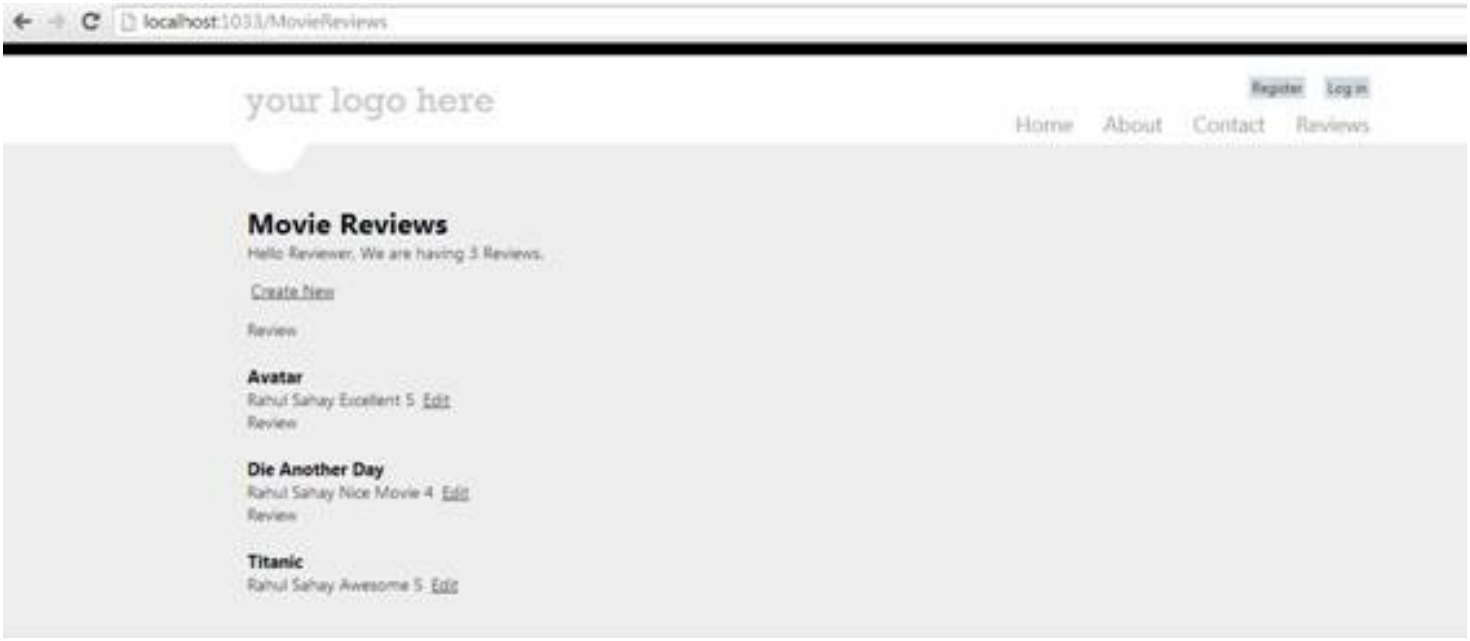
```
//
        // POST: /MovieReviews/Edit/5

        [HttpPost]
        public ActionResult Edit(int id, FormCollection collection)
        {
            try
            {
                var movieReview = _movieReviews.Single(m => m.Id == id);
                if(TryUpdateModel(movieReview))
                {
                    //To Do:- database code
                    return RedirectToAction("Index");
                }
                return View(movieReview);
            }
            catch
            {
                return View();
            }
        }
```

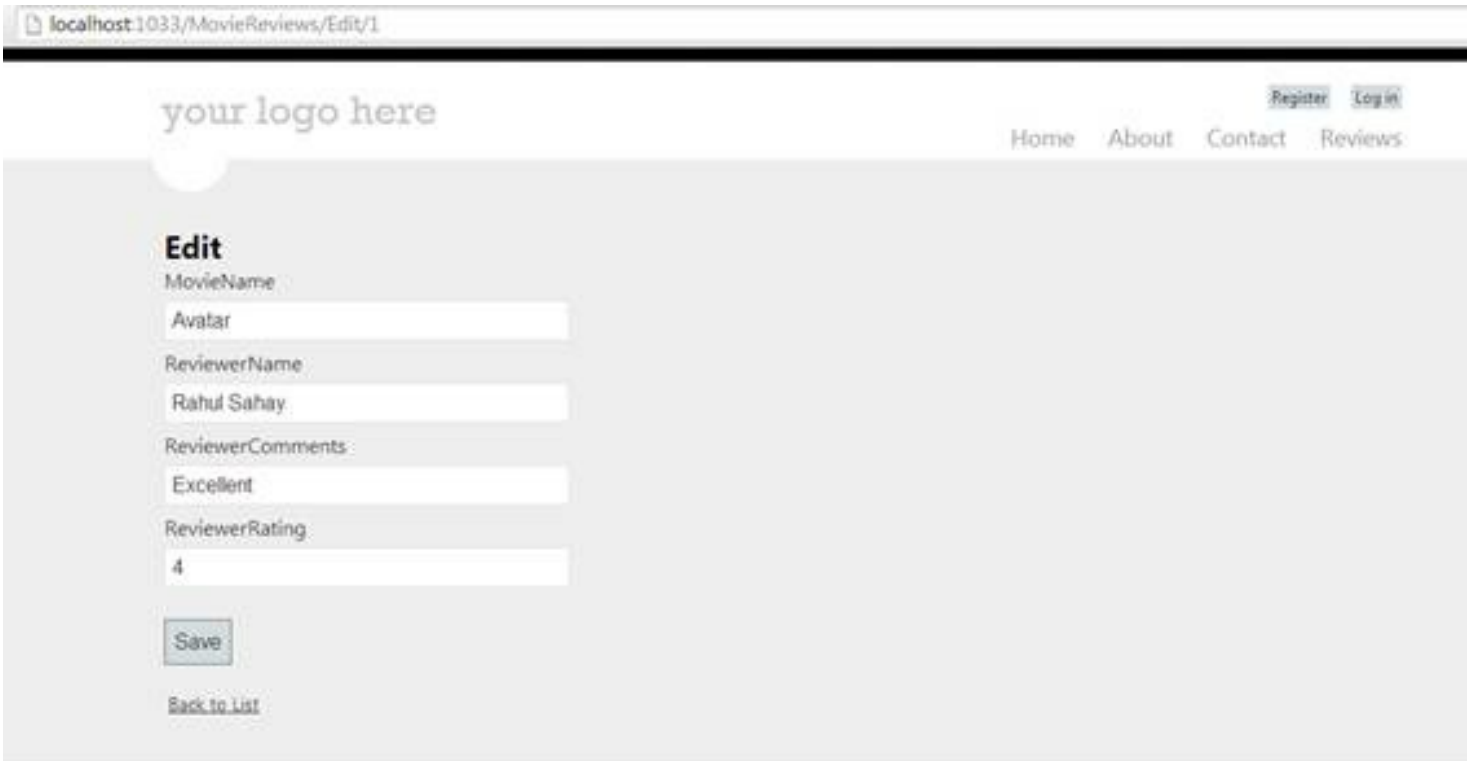
So, when I say "***TryUpdateModel(***movieReview***)***", the model binder will go and look at the **movieReview**, look for **rating property** whether it has one. Now, there should be a posted form input named rating, so the MVC runtime will find that and just

move it in for me. So, during this entire course of action if anything fails, if any validation error occurs, "***TryUpdateModel***" will return false and then we don't save the review.

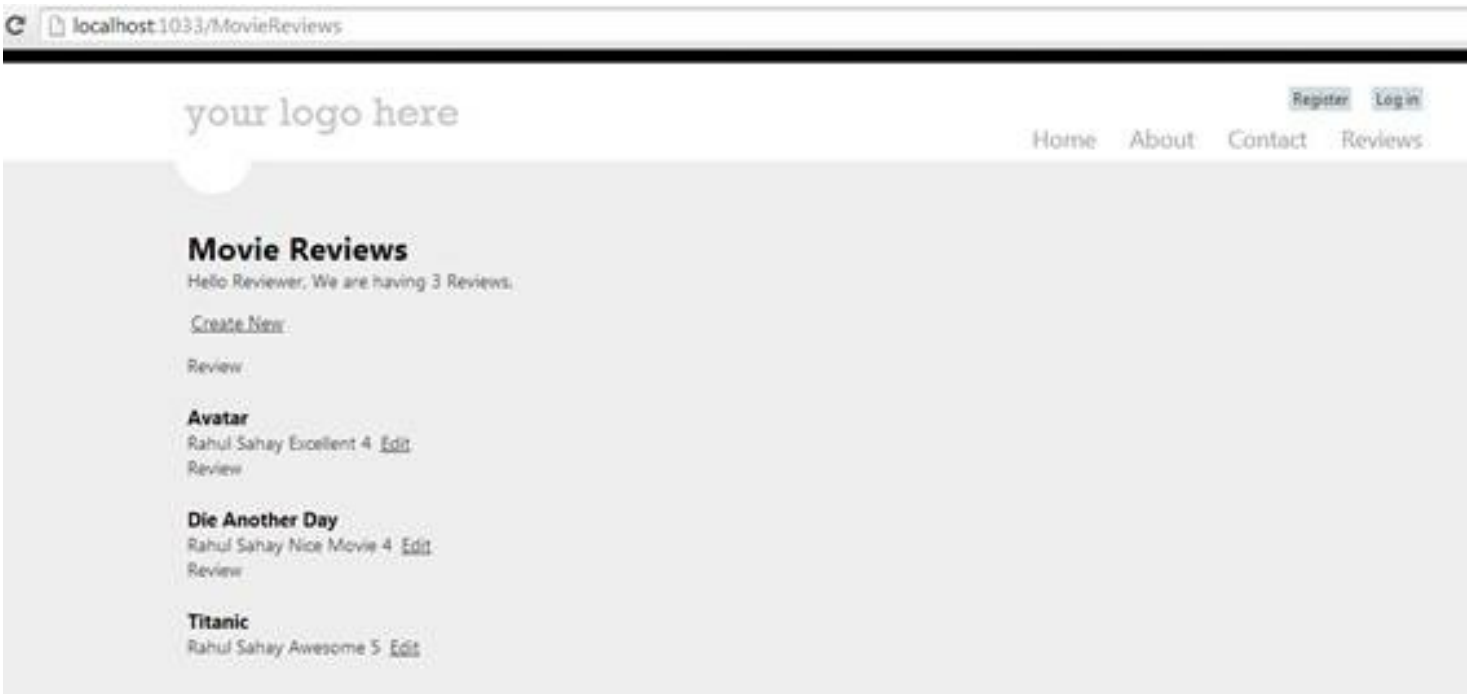
We'll do deep detailing about validation in the coming chapter. So, if that fails then I just simply return the view with the same old **movieReview** to the user to fix the same and submit again. But, if everything goes fine, at that instant we want to save the review to the database. But, at this instant we are not working with database, hence just return it to index page. Now, when I build and run the app, then this will produce me the below result.



Now, when I click the edit link, it will produce me the below result.

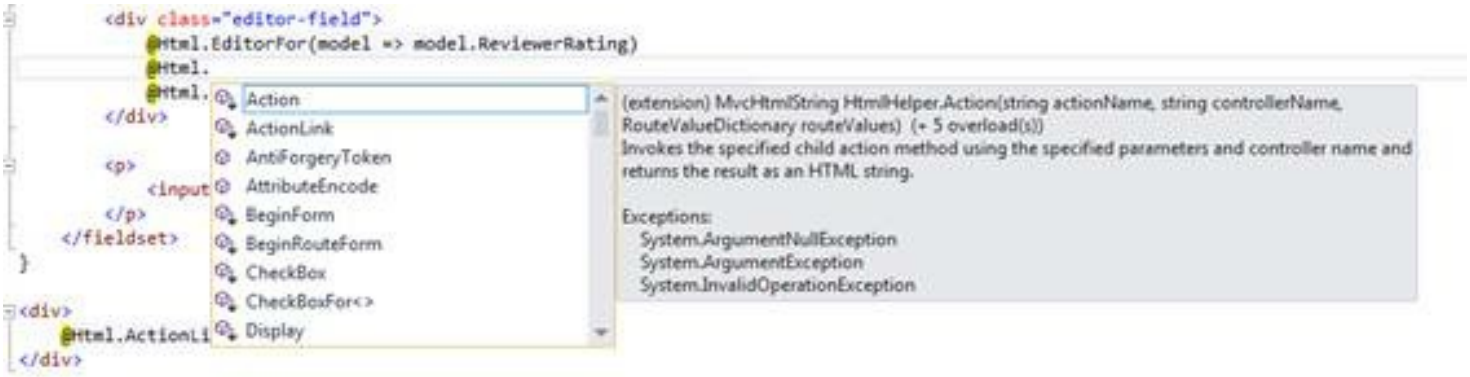


So, in the above screen shot, I have changed the rating to 4, and then when I save the form, it will redirect me to the index page of the Movie Review with the modified value.



Please note that these changed values are not going to commit in the database, these are in memory static values, so when you closed the browser, these changed values will get lost. Apart from

these HTML Helpers, there are many other HTML Helpers which you could see in intellisense window and its usage.



Now, one important point to note here that every HTML property inherits from its base class; and this HTML property of type "***System.Web.Mvc.HtmlHelper<T>***", where T is a generic type parameter representing the type of the model for the view (dynamic by default). So, as we see in the above screenshot, all these extension methods are at your help for

building MVC Application. You should only know its implementation and when to use the same. However, if you think these generic extension methods are not going to serve your purpose, you can obviously build your own extension methods.

HTML.ValidationSummary:-

The Validation Summary property displays an un-ordered list of all validation errors in the "***ModelState***" dictionary. Now, the Boolean parameter which we are using (with a default value true) is telling the helper to exclude property-level errors; which means you

are telling the summary to display only the errors in "***ModelState***" associated with the model itself, and exclude any errors associated with a specific model property. For instance you have the following code somewhere in the controller action rendering the edit view:

```
"ModelState.AddModelError("", "Without Property");"
"ModelState.AddModelError("MovieName", "With Name Property!");"
```

So, the first error is a model-level error, because we didn't provide a key to associate the error with a specific property. The second error we associated with the "***MovieName***" property, Hence in your view it will not display in the validation summary area

(unless you remove the parameter from the helper method, or change the value to false). In this scenario, the helper renders the following HTML:

```
<div class="validation-summary-errors">
<ul>
<li>Without Property</li>
</ul>
</div>
```

Some more HTML Helpers:-

So, in this case we 1st look at "***HTML.TextBox***" and "***HTML.TextArea***". So, the textbox helper renders an input tag with attribute "***text***". Basically, we use this to accept

any kind of inputs from users; For instance

"@Html.TextBox("MovieName", Model.MovieName)"

This produces below output, when you see the source of the page rendered.

"<input id="MovieName" name="MovieName" type="text" value="Avatar" />"

Also, related to the above helper we have "***TextArea***". We use this helper when we want user to enter multi line description, like movie Review or in more general Address.

The following code:

*"@Html.TextArea("text", "Hello
 MVC")"*

Produces:

*"<textarea cols="20" id="text" name="text" rows="2">Hello
 MVC </textarea>"*

HTML.DropDownList and HTML.ListBox

Now, both dropdownlist and ListBox helpers return a "**<select/>**" element. Now, here "**DropDownList**" returns the single item whereas "ListBox" returns for multiple items.

Here, it requires a bit of work in the controller. A list needs a collection of "**SelectListItem**" instance representing all the possible entries for the list. Also, a "**SelectListItem**" object will have a "**Text**", "**Value**", and **Selected**

properties. We can also build the collection of "**SelectListItem**" objects our self, or rely on the "**SelectList or MultiSelectList**" helper classes. These classes can look at an "**IEnumerable**" of any type and transform the sequence into a sequence of "**SelectListItem**" objects. For instance, we can have one sample as shown below

```
public ActionResult Edit (int id)
{
var movie = moviedb.Movies.Single(m => m.MovieId == Id);
ViewBag.DirectorName = new SelectList(moviedb. DirectorName.OrderBy(g => g. DirectorName),
"MovieId", "DirectorName", movie. MovieId);
return View(album);
}
```

You can think of the controller action as building the primary model (the album for editing) and also the Presentation model required by the drop-down list

helper. Here, the parameters to the "***SelectList***" constructor specify the original collection, the name of the property to use as a value ("***MovieId***"), the name of the property to use as the text (Name), and the value of the currently selected item (to determine which item to mark as selected).

Now, the same thing can be done other way as well, If you want to avoid reflection overhead and generate the "***SelectListItem***" collection yourself, you can use the LINQ Select method to project Movie into "***SelectListItem***" objects:

```
public ActionResult Edit(int id)
{
var movie = moviedb.Movies.Single(a => a.MovieId == Id);
ViewBag.DirectorName =
moviedb.Movies
.OrderBy(g => g. DirectorName)
. AsEnumerable()
.Select(m => new SelectListItem
{
Text = m. DirectorName,
Value = g. DirectorName,
Selected = movie.MovieId == m.MovieId
});
return View(movie);
}
```

Html.ValidationMessage

Whenever there is an error for a particular field in the "***ModelState***" dictionary, you can use the

"***ValidationMessage***" helper to display the error message. For instance, in the following controller action, we purposely add an error to model state for the "***MovieName***" property:

```
[HttpPost]
public ActionResult Edit(int id, FormCollection collection)
{ var movie = moviedb.Movies.Find(id);
ModelState.AddModelError("MovieName", "Displaying Error message for Movie Title");
return View(movie);
}
```

Now, in the view, you can display the error message (if any) with the below code:

@Html.ValidationMessage("MovieName

which will produce:

```
<span class="field-validation-error" data-valmsg-for="Title" data-valmsg-replace="true">
Displaying Error message for Movie Title
</span>
```

This message appears only if there is any error in the model state for the key "***MovieName***".

HTML.Password

Now, the "***Html.Password***" helper renders a password field which is in nature much like the "***TextBox helper***", except that it does not retain the posted value rather it uses a password mask. The following code:

```
@Html.Password("YourPassword")
```

Produces:

```
<input id="YourPassword" name=" YourPassword" type="password" value="" />
```

The strongly typed syntax for Html.Password, as you'd expect, is "***Html.PasswordFor***". Here's how you will use it to display the "***YourPassword***" property:

```
@Html.PasswordFor(m => m.YourPassword)
```

HTML.RadioButton

Radio buttons are generally grouped together to provide a range of possible options for a single value. For instance, if you want the user to select a programming language from a specific list of languages, then you can use multiple radio buttons to present the options.

The "***Html.RadioButton***" helper renders a simple radio button. Below we have few samples explaining the same.

```
@Html.RadioButton("language", "C#", true)
@Html.RadioButton("language ", "VB")
@Html.RadioButton("language ", "C++")
```

This will produce the below result:

```
<input id=" language" name=" language" type="radio" value=" C#" checked="checked" />
<input id=" language" name=" language" type="radio" value="VB" />
<input id=" language" name=" language" type="radio" value="C++" />
```

Html.RadioButton has a strongly typed substitution as well, "***Html.RadioButtonFor.***Now, strongly typed version takes an expression that identifies the object which contains the property to render, followed by a value

to submit when the user selects the radio button.

```
@Html.RadioButtonFor(m => m.MovieId, "1") Avatar
@Html.RadioButtonFor(m => m.MovieId, "2") Titanic
@Html.RadioButtonFor(m => m.MovieId, "3") Godzilla
```

HTML.CheckBox

Now, comes is checkbox helper which is unique in many aspects because it renders two input elements. For instance

```
@Html.CheckBox("IsReady")
```

This code produces the following HTML:

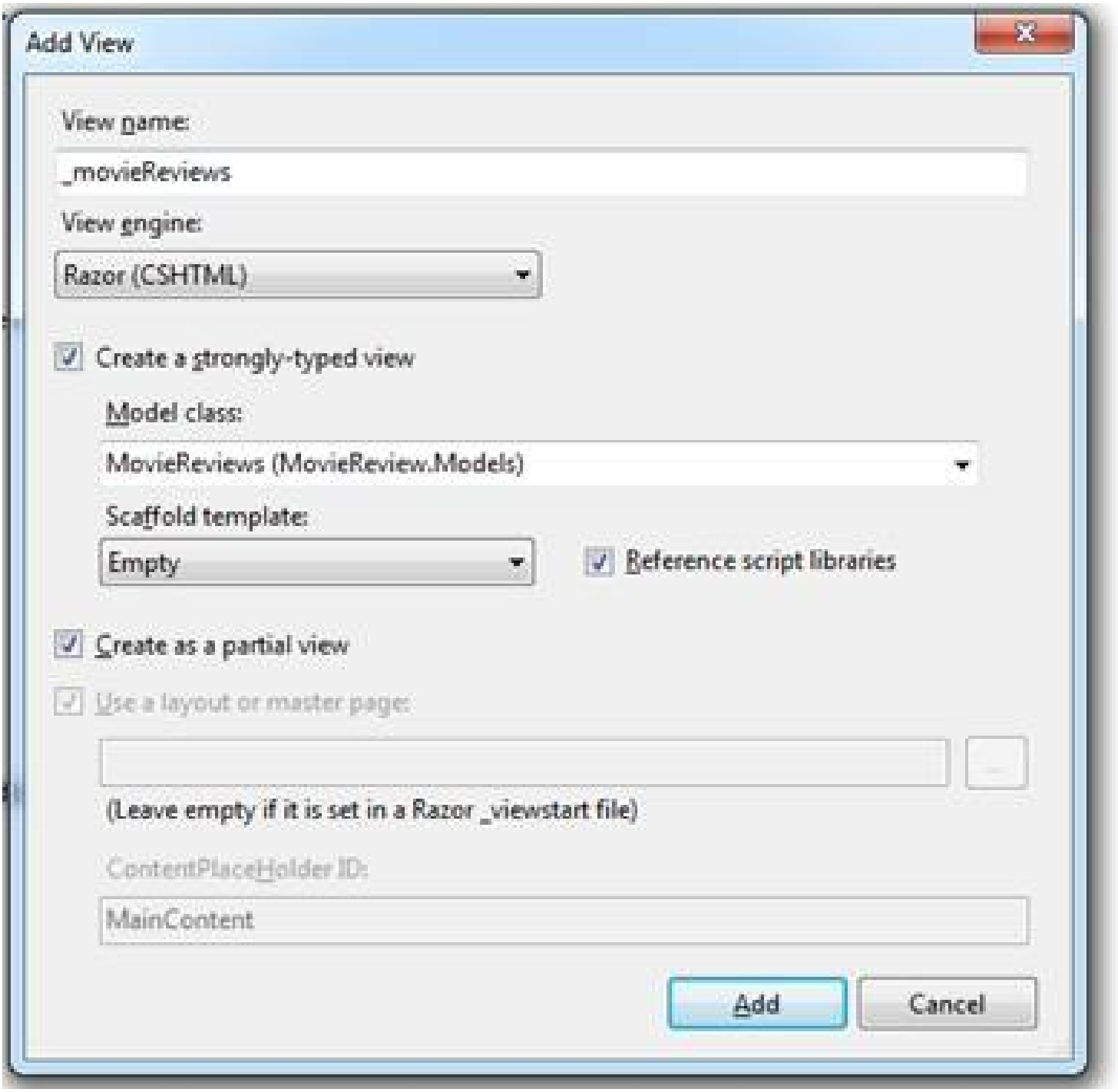
```
<input id="IsReady" name="IsReady" type="checkbox" value="true" />
<input name="IsReady" type="hidden" value="false" />
```

Now, here you might be thinking that why it renders two inputs one with hidden behavior. Now, one important point to note here is that HTML specifications indicate that a browser will submit a value for checkbox only when the checkbox is selected. Now, here the second input guarantees a value will appear for "**IsReady**" even when the user does not check the checkbox input.

Partial Views:-

Partial Views are just another type of view which enhances reusability. So, think of partial views like user controls

where in you have some common pieces of code which needs to be reused time and again at various places. So, in my application example let me cut a piece of code which I am going to render at different places and create one partial view.



Above is the screen shot, for creating partial view, I have named the same starting with "_". Well underscore (_) is just a convention for naming any partial

view, it just depends on you, and how you name partial views. I just do the same in order to identify partial views by just looking it. You might have noticed while creating the same, I have used the same model which I have used earlier but this time I have selected the empty template from the scaffolding option, because I am already having the markup which my partial view is going to have.

Now, upon clicking add button partial view gets created. Now, in here I have copied the markup from the index page of the movie review and just replaced the term **item** with **Model**, because here in partial view, we'll be taking care of

one movie at a time.

```
@model MovieReview.Models.MovieReviews
```

```
<div>
    <h4>@Model.MovieName</h4>
    <span>@Model.ReviewerName</span>
    <span>@Model.ReviewerComments</span>
    <span>@Model.ReviewerRating</span>
    <span>@Html.ActionLink("Edit","Edit",new{Id=Model.Id})</span>
</div>
```

Now, in order to render the partial view from the Index page, I just need to do one small thing as shown below.

```
@model IEnumerable<MovieReview.Models.MovieReviews>

@{
    ViewBag.Title = "Movie Reviews";

}

<h2>Movie Reviews</h2>

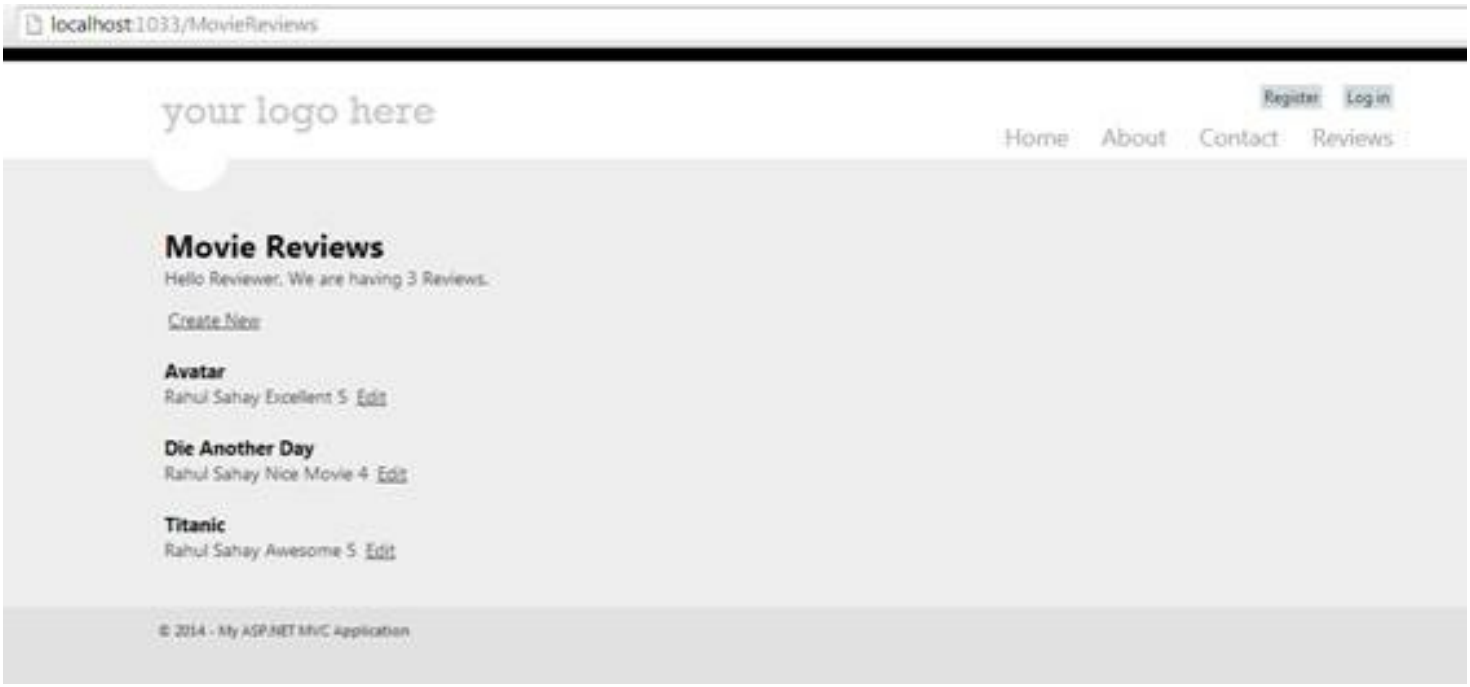
    Hello Reviewer, We are having @Model.Count() Reviews.

<p>
    @Html.ActionLink("Create New", "Create")
</p>

@foreach (var item in Model)
{
    @Html.Partial("_movieReviews", item)
}
```

So, in here I have just used HTML helper to render the partial view. I have used its name with the individual model to pass in the properties required. With this change in place, when I save everything and refresh the page, it will

return me the same output, but this time with the partial view.



Let me explain one more scenario where in you want to use partial view. You may land up in a situation where in you want to render something and that is not part of the model, it generally happens with the layout views.

Since layout is application wide; hence

so it's difficult to restrict the layout view with any model. So, let me consider a scenario where in you always want to show best movie reviews at the bottom of every page. Now, in order to do that, you need to implement the same on the layout page.

So, in order to fulfil the requirement, we'll be using one more HTML Helper called as "***@Html.Action***". Now, this "***@Html.Action***" will setup a sub request inside the primary MVC Request. So, it's basically a sub request that goes out and call another controller action which builds its own model dependent on what main controller is doing and render its own partial view.

So, in the layout page I'll add this piece of code.

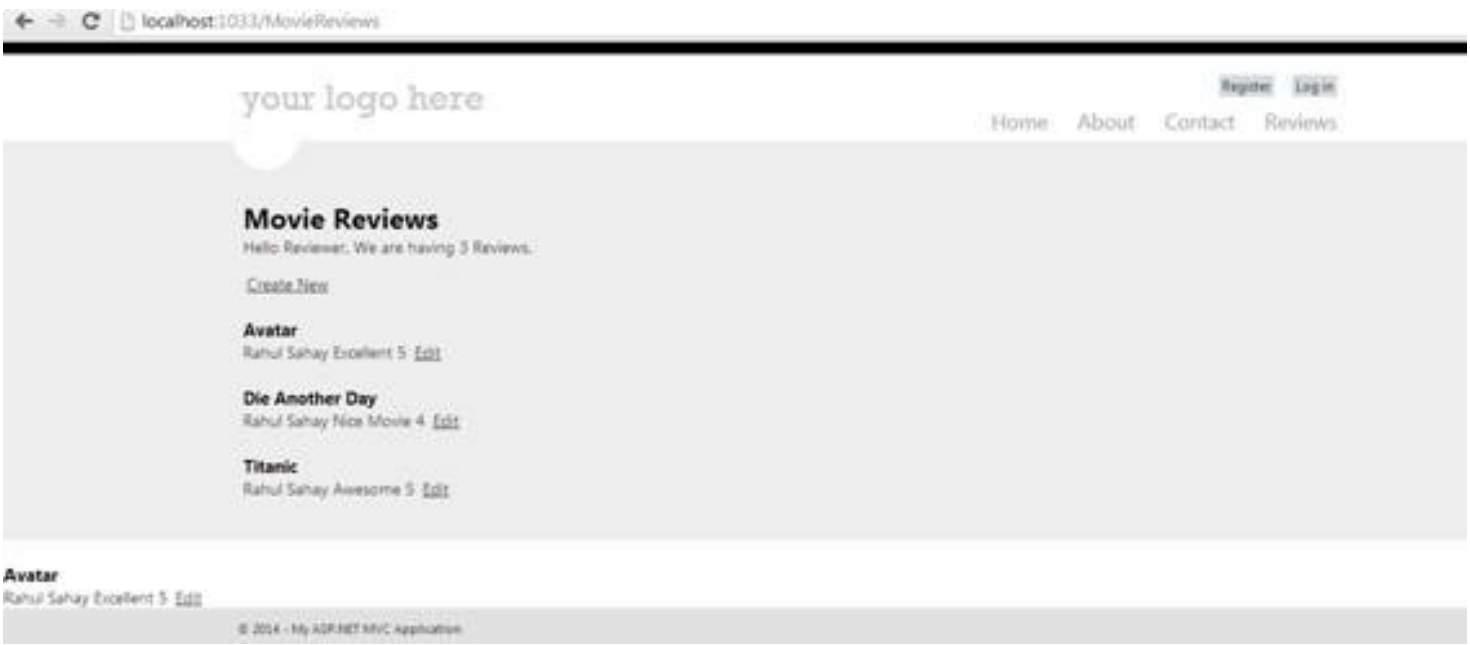
```
<div>
            @Html.Action("bestReviews","MovieReviews")
        </div>
```

So, up here I am looking for the action method "***bestReviews***" in the "***MovieReviews***" controller. Now, let me go ahead and create the desired Action method.

```
public ActionResult bestReviews()
        {
            var movieReview = from m in _movieReviews
                              orderby m.ReviewerRating descending
                              select m;
            return PartialView("_movieReviews", movieReview.First());
        }
```

Now, up here what I am doing is, I have created one action method, which is

returning partial view which I have already created with the required model. In order to construct the model, I have just used a little LINQ query to fetch the best rated sorted movie. Now, when I build the same refresh the page, it will present me the below output.



Viewbag and Viewdata:-

Last but not the least; I thought to cover this topic in this Views chapter only. You might have noticed that while creating any view and passing the data from the controller, there is one property with the name "***ViewBag***" which is getting set like

```
public ActionResult Index()
        {
            var Controller = RouteData.Values["controller"];
            var Action = RouteData.Values["action"];
            var Id = RouteData.Values["id"];

            string Output = string.Format("Controller = {0}, Action= {1}, Id= {2}",
Controller, Action, Id);
            ViewBag.Message = Output;

            return View();
        }
@{
    ViewBag.Title = "Home Page";
}
```

So, what is happening here, that data is getting passed in from controller to view via "***ViewDataDictionary***", a special

dictionary to maintain "***ViewData***" . However, if you remember, that you can set and get property from the standard dictionary property like ViewData["CurrentDay"]= DateTime.Now.Day

Now, from the definition point of view, "***ViewBag***" is just a dynamic wrapper around "***ViewData***". So, the above example could also be written like ViewBag.CurrentDay= DateTime.Now.Day

So, from my point of view I didn't see any technical advantage, one is getting over other, it depends on one perspective, that how they are willing to

choose one. It's all depending on developer's friendliness with one.

Summary:-

So, in this section we have seen Views and their usage in Detail. We have touch base on different kind of scaffolding templates, while creating the views. We have also seen that how to prevent the XSS attack. Then, we have seen how to use HTML Helpers in various cases. We have covered multiple important HTML helpers and it's usage in brief. We have also seen Layout page and its usage, like we have created multiple layout pages in our application. We have also used

partial views to enhance the reusability. In the end we have covered ViewData and ViewBag property with a simple example.

Chapter 4: Getting Started with Data Modeling

WHAT DO you find in this CHAPTER?

⇒ Introduction

⇒ Using Entity Framework

⇒ Entities Creation

⇒ Database Migrations

⇒ LINQ

⇒ Search Query

⇒ Summary

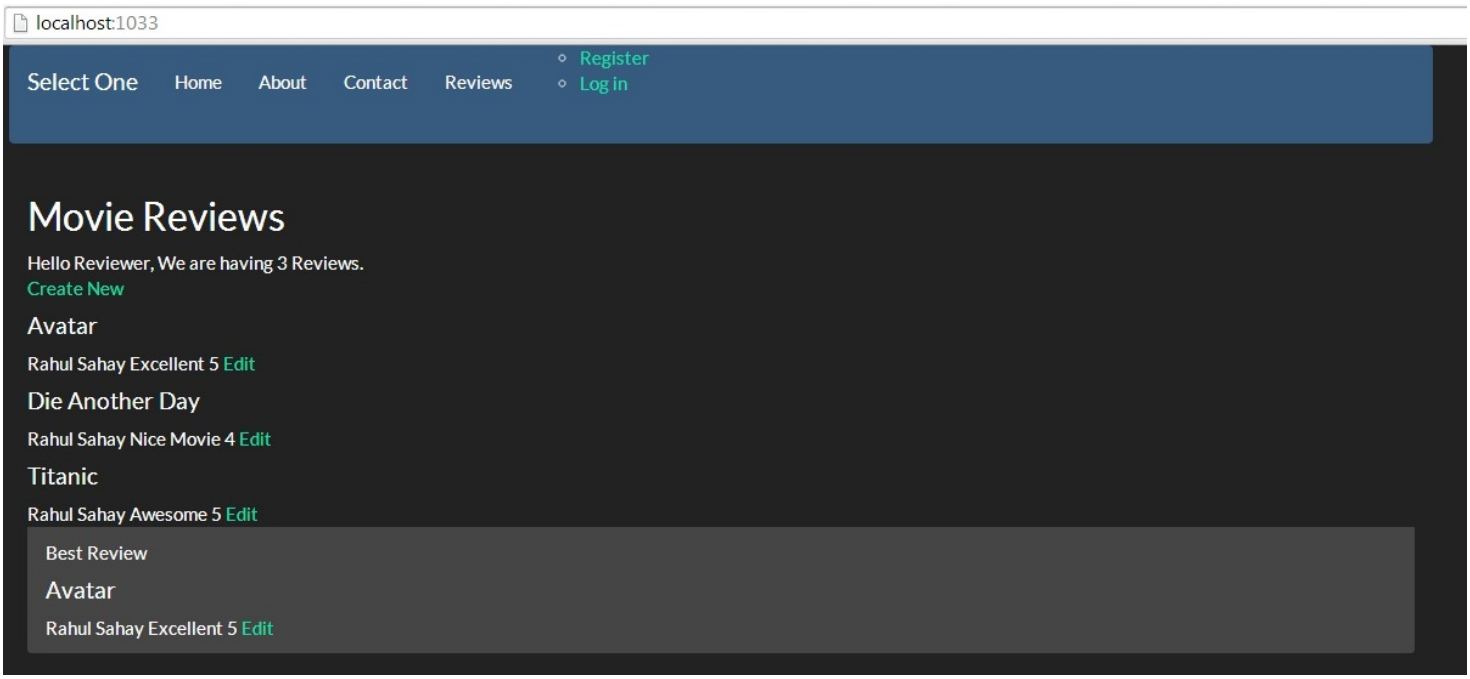
Introduction:-

Hi Friends, I hope you have enjoyed the journey towards learning MVC. In this entire course having said to my earlier

statement, my entire focus is on building End To End application, so that at the end of the day, you not only grasp the theoretical concepts rather understand their usage practically as well. So, in this section our entire focus will be on building **Data Model**. In this case, we'll look one interesting technology for building the same known as "***EntityFramework***" also known as **EF** and then combine the EF with plain simple C# class definition to build an application.

We'll also use **SQL Server** to persist the data and **LINQ** to query against the database to fetch the records. But, before jumping to build data model, let me

introduce you with the new UI of our application. I'll just do touch base on UI features and improvements as the scope of this book is MVC only.



So, as you see in the above screen shot, the **Layout** and **Theme** has been changed. I have used **Bootstrap Templates** to boost my **UI** design. The idea behind these templates is, these templates are lightweight, easy to use,

mobile friendly, adaptive to HTML 5 features, easily adjustable to any devices of different resolutions. So, you can download any of these free templates from (***http://bootswatch.com/***) and then integrate the same in your application.

Using Entity Framework:-

In this section, the 1st step which I am going to focus is on setting up **Entity Framework**. Since, EF is part of **.NET Framework** and we already have a reference of EF in MVC Internet application. Basically, EF allows me to access relational database using strongly typed **C# LINQ** Queries. So, the

reasoning over here is that when I am working with classes to talk to EF, I really don't have to worry about legacy SQL settings like (***Sql Connections***, ***Sql Commands***, ***Sql Parameters*** etc.). So, these were the basic things which we used to program with SQL Server using "***ADO.NET***" prior to EF. So, with the EF, I can directly work with C# code and I can issue query against the database using the "***LINQ (Language Integrated Query)***" technology. Primarily there are three different styles to get started with EF. I have a very detailed explanation of EF in my blog which you can refer @ (***http://myview.rahulnivi.net/?cat=1***) . But, let me give brief snapshot about

different approaches.

Schema First Approach:- So, basically in this approach what we do, we open a graphical designer in Visual Studio where in we are pointing the EF to an existing database so that we can import the database schema. Now, this will import the entire schema and generate the classes which we need to query and update the database back.

Model First Approach: - In this approach I use the same graphical designer inside the visual studio, to draw an overall model of my application. And then EF will generate both my class definitions and my

database schema.

Code First Approach: - Last but not the least there is one **Code First Approach**. This one is my personal favorite. In this case, I just write C# classes and then EF use the class definitions to create the database for me. And for doing the same, EF follows certain conventions like Naming Conventions. We can also provide explicit mappings wherein I can change the mappings if I don't like the default one. So, in this book, I'll be using Code First Approach.

As a FYI, let me tell you that EF doesn't mandate any kind of database what you have in the backend. It works well with

variety of different technologies like

Sql Server
Oracle
DB2
Cloud etc…

Now. In this case we are working with **SQL Server**. As I said, I have just put the definitions of different EF style; But, in order to see them in action, how to get started with them, and use it in your application, refer my blog wherein I have written complete set of examples (***http://myview.rahulnivi.net/?cat=1***) explaining each style in detail.

Entities Creation:-

Now, the time has come to setup the environment in which data can persist. So far we have been using in memory data to test the application. Now, from here I also need to move few properties from "***MovieReview***" class to a new class say "***Movie***" class and then build the relationship between these two classes.

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Web;

namespace MovieReview.Models
{
    public class Movie
    {
        public int Id { get; set; }
        public string MovieName { get; set; }
        public string DirectorName { get; set; }
        public string ReleaseYear { get; set; }

        public ICollection<MovieReviews> reviews { get; set; }
    }
}
```

And then, every movie class will be using some associated Review. So, I have put the same in "***ICollection***", a collection of Movie Reviews. So, this is how I am building the relationship between the **Entities**. Now, I can go

ahead and change the original "***MovieReviews***" class a bit.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace MovieReview.Models
{
    public class MovieReviews
    {
        public int Id { get; set; }
        public string ReviewerName { get; set; }
        public string ReviewerComments { get; set; }
        public int ReviewerRating { get; set; }
        public int MovieId { get; set; }
    }
}
```

However, if you have noticed in the above class definition, I have just removed the "***MovieName***" from the class and added one property with the name "***MovieId***". So, here "***MovieId***" field is pointing back to the Movie class with which review is associated. Now, these two classes are going to be my entities which I am going to use to save the records in the database and retrieve the same from the database. In actual scenario means in the enterprise level, I always prefer to have a class library project, basically an independent project for my model classes just to increase the reusability and isolation level to a greater extent. But, in this case I am leaving the same in the Models folder

which Visual Studio has created for me by default. Now, the next step is going to add another class which is going to talk to EF, to retrieve and save the data for me.

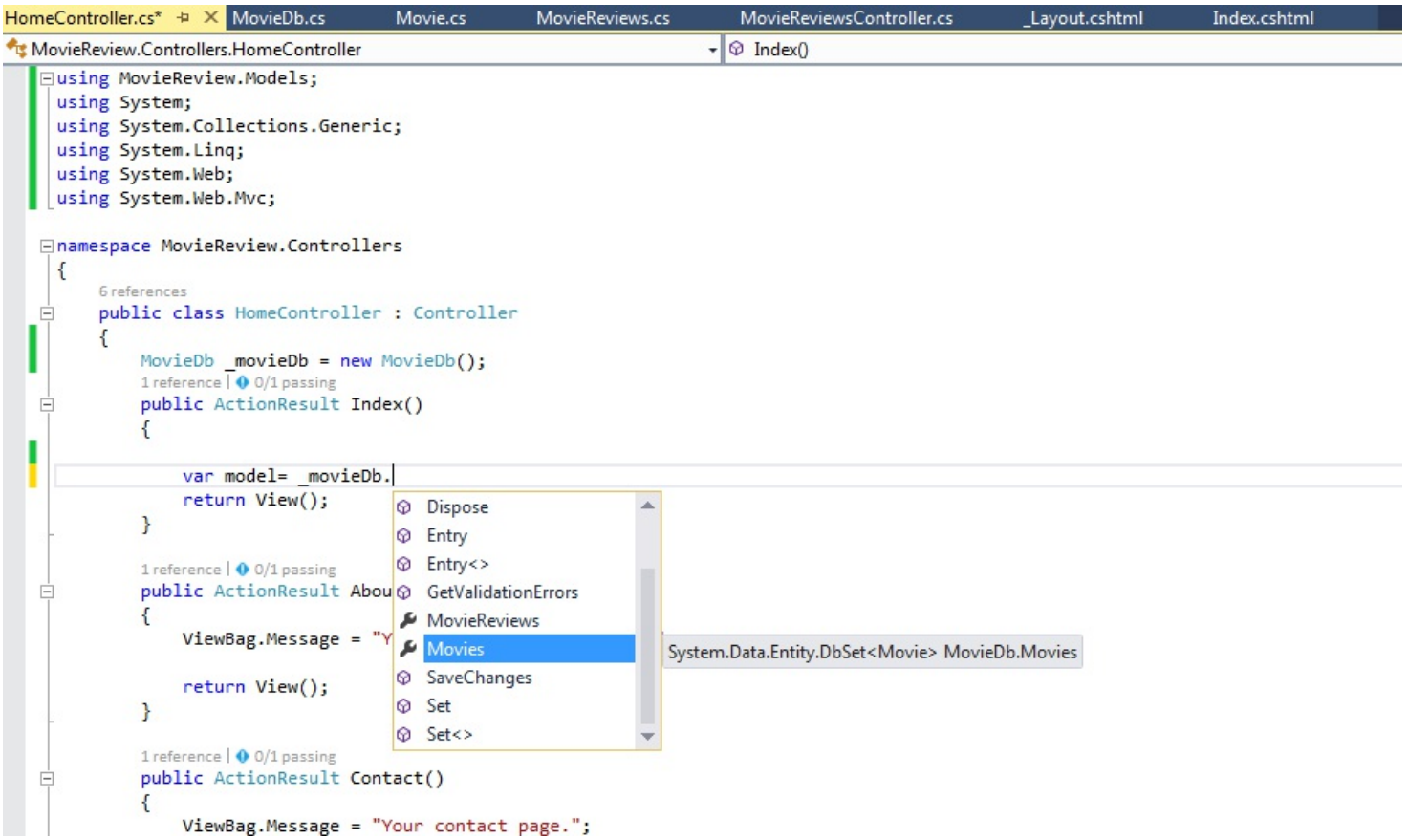
```
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Web;

namespace MovieReview.Models
{
    public class MovieDb:DbContext
    {
        public DbSet<Movie> Movies { get; set; }
        public DbSet<MovieReviews> MovieReviews { get; set; }
    }
}
```

As, you can see in the above snippet,

"***MovieDb***" derives from an EF class known as "***DbContext***". And then, in that DbContext, I have used couple of properties with the name "***DbSet***", which basically represents the entity which you want to query and save.

Now, let me go ahead and use the above definition to bring in some data and produce the same on the home page. So, for doing the same I need to modify the code a bit in the Home controller 1st. So, let's do that.



Now, in the above screen shot, you can see EF in action, it has listed my both "***DbSets***", with which I am going to fetch the data and persist the same back to database.

```
using MovieReview.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace MovieReview.Controllers
{
   public class HomeController : Controller
   {
      MovieDb _movieDb = new MovieDb();
      public ActionResult Index()
      {

         var model = _movieDb.Movies.ToList();
         return View(model);
      }

      public ActionResult About()
      {
         ViewBag.Message = "Your app description page.";

        return View();
      }

      public ActionResult Contact()
      {
         ViewBag.Message = "Your contact page.";

         return View();
      }

      protected override void Dispose(bool disposing)
      {
         if (_movieDb != null)
         {
            _movieDb.Dispose();
         }
         base.Dispose(disposing);
```

```
        }
    }
}
```

With the above piece of code, what the EF will do, this will go in the SQL Server and find all the movies from the database and present them in the list. Now, since this connection is a disposable resource, hence I need to override the dispose method and tell it that if that is not null, dispose the same. Thus, the dispose API in .Net is just a way to clean things up for you.

Now, once you go ahead and do build here, it will obviously throw couple of compiler error and the reason being for

it that we have modified the class definitions. So, for the time being you can just go ahead and comment out the "***MovieReviews***" Controller code completely. There is one more change you need to do is in the Layout view where in you are displaying the Best Reviews, comment out this as well.

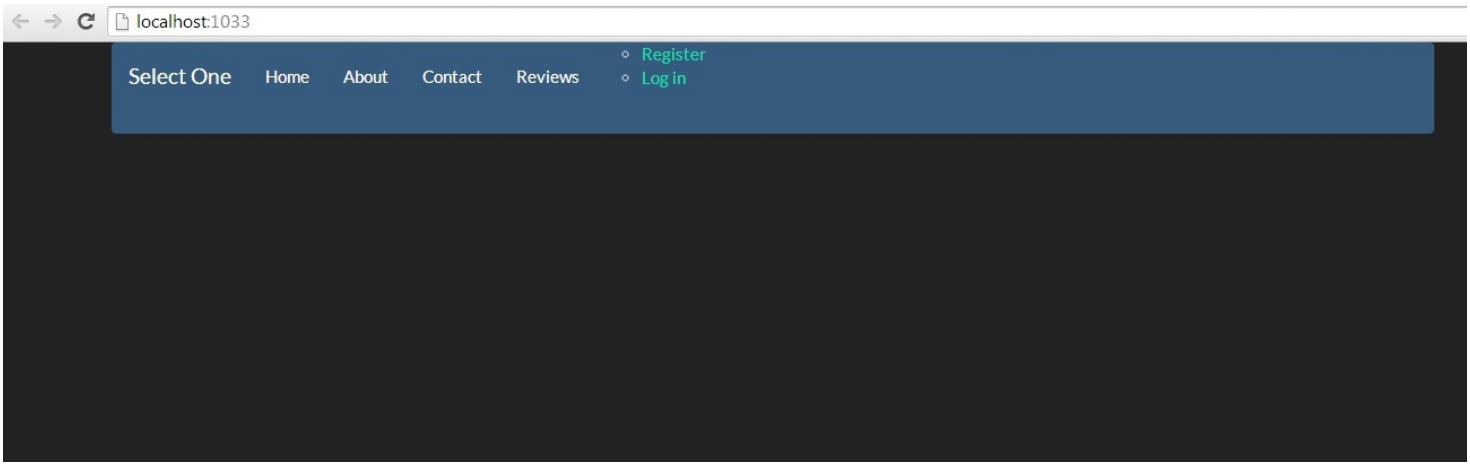
Now, once the controller methods are done, next step is to fix the view. I'm already having the view created, just need to modify the same so that it will be bound to strongly typed movie class. Below, is the modified Index view for the same.

```
@model IEnumerable<MovieReview.Models.Movie>
@{
    ViewBag.Title = "Home Page";
}

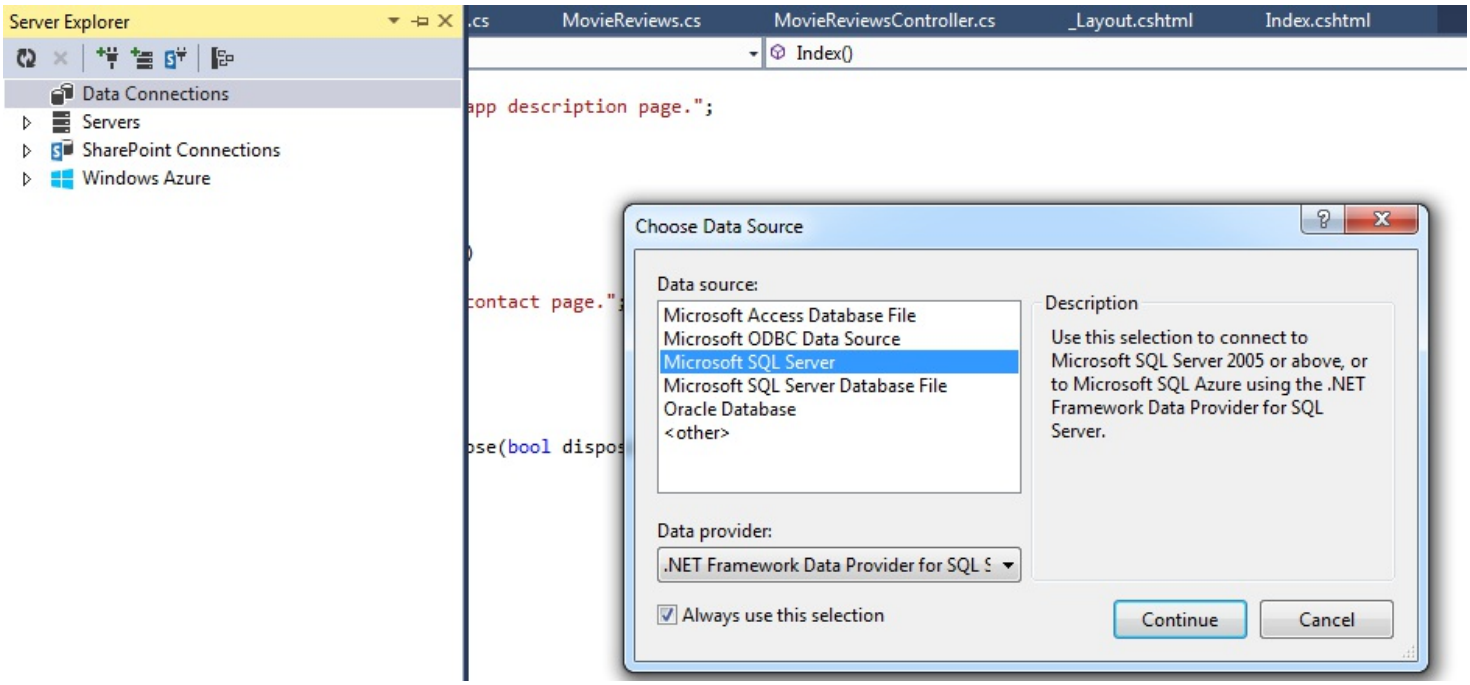

@foreach (var item in Model)
{
    <div>
        <h4>@item.MovieName</h4><br />
        <h5>@item.DirectorName</h5>
        <h5>@item.ReleaseYear</h5>
        <hr />
    </div>
}
```

But, at this point of time I don't have database associated yet. But, let me build and run the app, and let's see the behavior of the application. So, as you

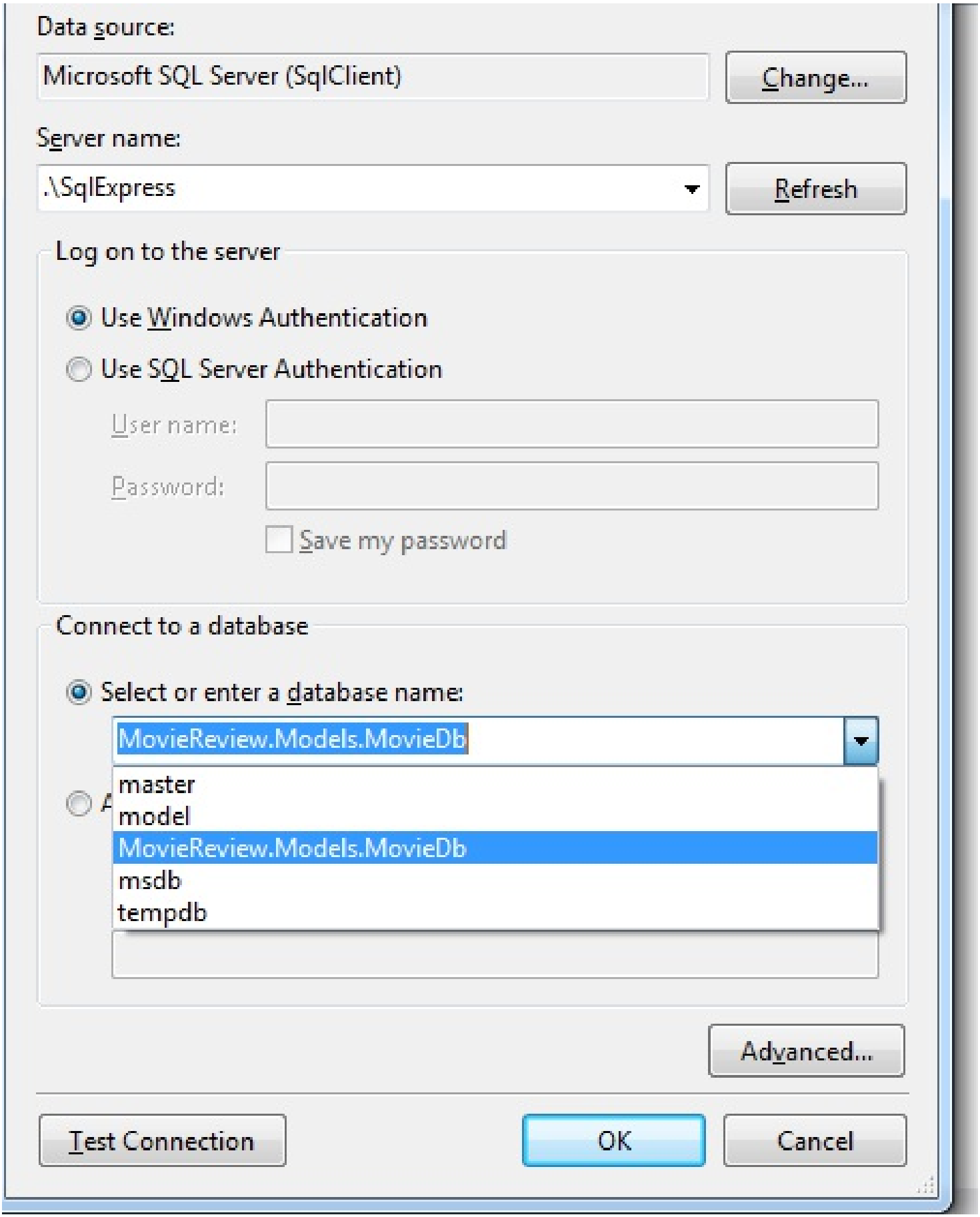
can see below in the screenshot, it just produced me the home page without data obviously, but didn't present any error. What was the reason for this?



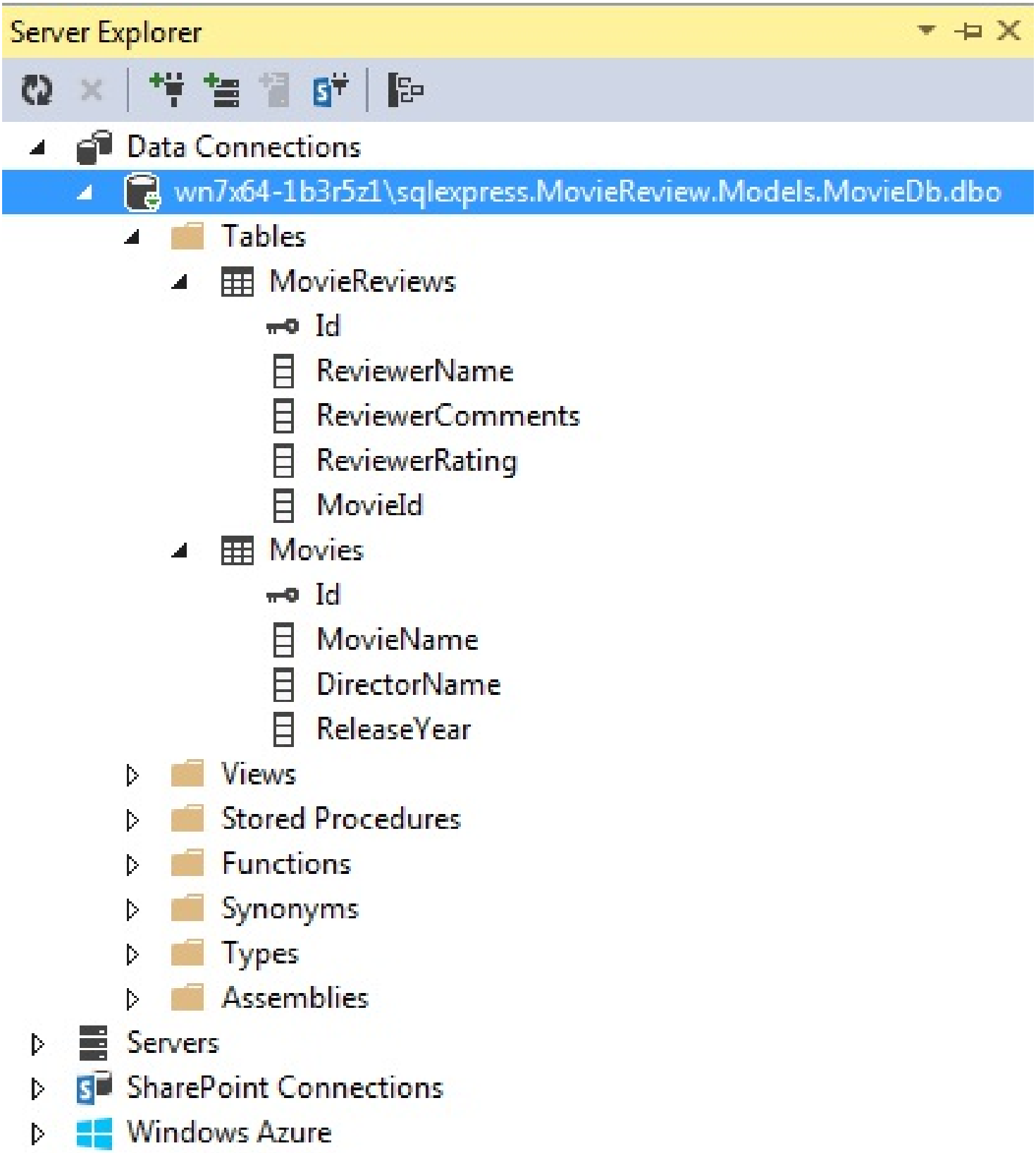
So, to anwser this question, let me go ahead and view the Server Explorer and add a connection.



This is going to be the SQL Server connection. And then I am going to specfiy the "***.\Sqlexpress***", as the server name which I am intrested in and then when I click on the databases dropdown, I can see that there is one database already created with the name"***MovieReview.Models.MovieDb***'.

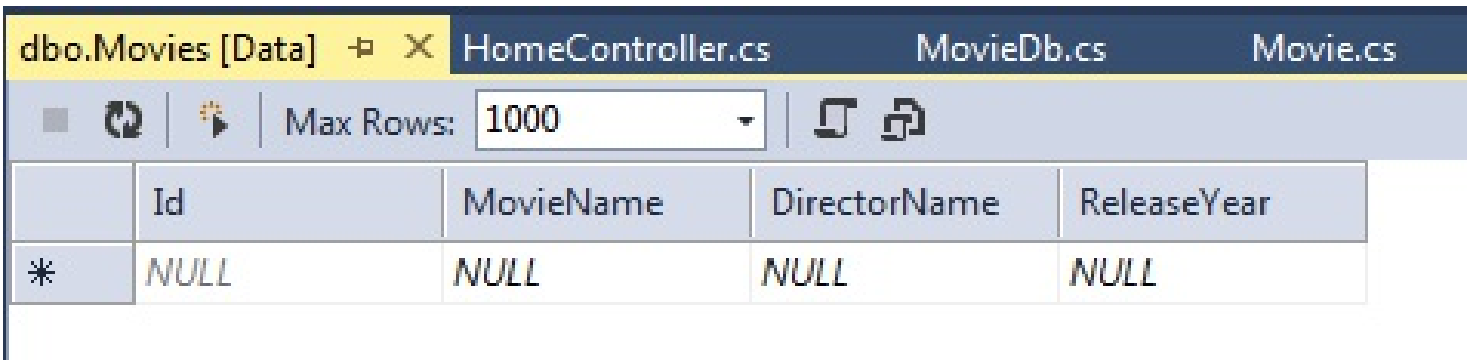

Data source:
Microsoft SQL Server (SqlClient)
Change...
Server name:
.\SqlExpress
Refresh
Log on to the server
Use Windows Authentication
Use SQL Server Authentication
User name:
Password:
Save my password
Connect to a database
Select or enter a database name:
MovieReview.Models.MovieDb
master
model
MovieReview.Models.MovieDb
msdb
tempdb
Advanced...
Test Connection
OK
Cancel

So, this is the database which EF created for me, because the name of the db matches the name with the class name which I have given. So, by default when you instantiate and query, and if **EF** doesn't find any database and you have no configurations done for the same, it will just create one with the class name in **SqlExpress**.



Now, in the above screen shot, you can

see the two tables which **EF** created based on classes which we have created. Now, if you do **ShowData** in one of the tables, that will obviously result null values, hence home page doesn't produce any result as such. But, to test this theory, let me go ahead and put some values in the table.

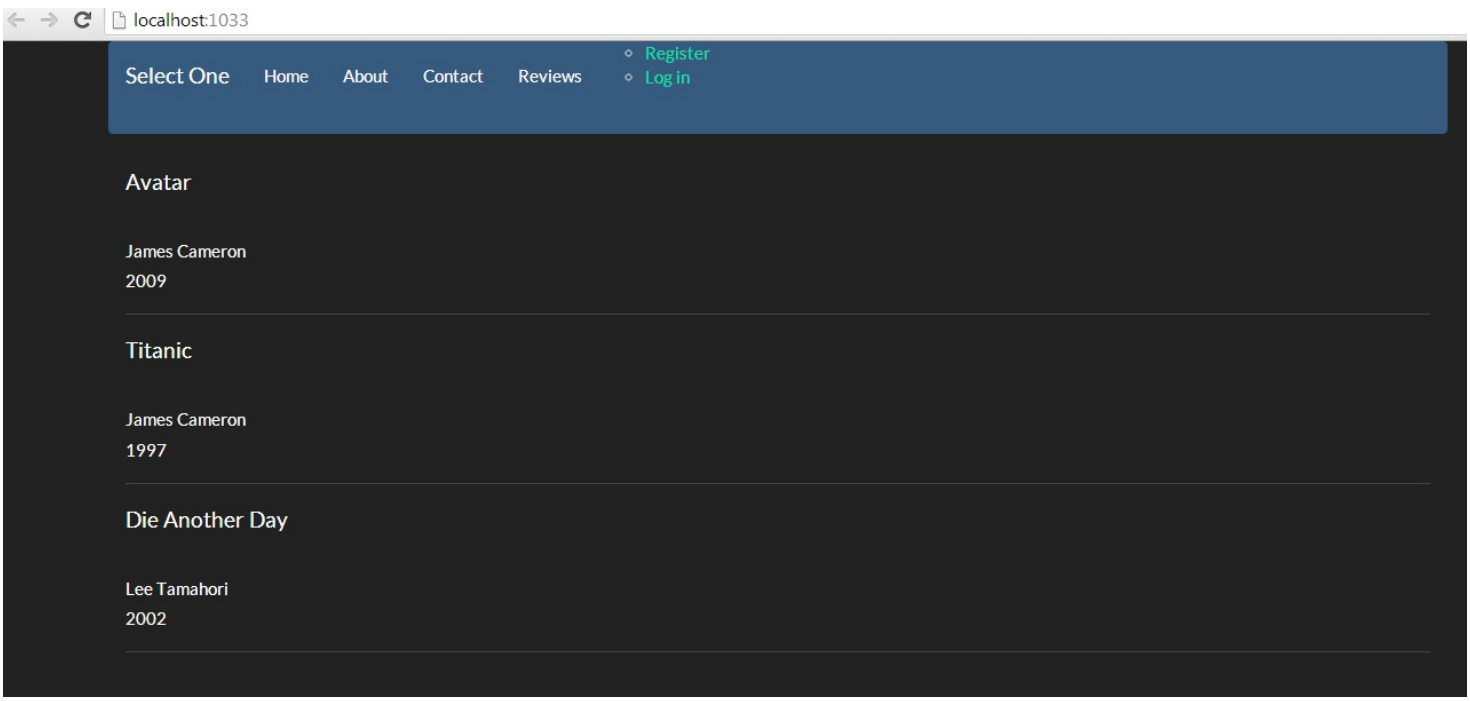


dbo.Movies [Data] | HomeController.cs | MovieDb.cs | Movie.cs

Max Rows: 1000

Id	MovieName	DirectorName	ReleaseYear
1	Avatar	James Cameron	2009
2	Titanic	James Cameron	1997
3	Die Another Day	Lee Tamahori	2002
NULL	NULL	NULL	NULL

Now, when you save the same and refresh the page. It should obviously produce me the desired result.

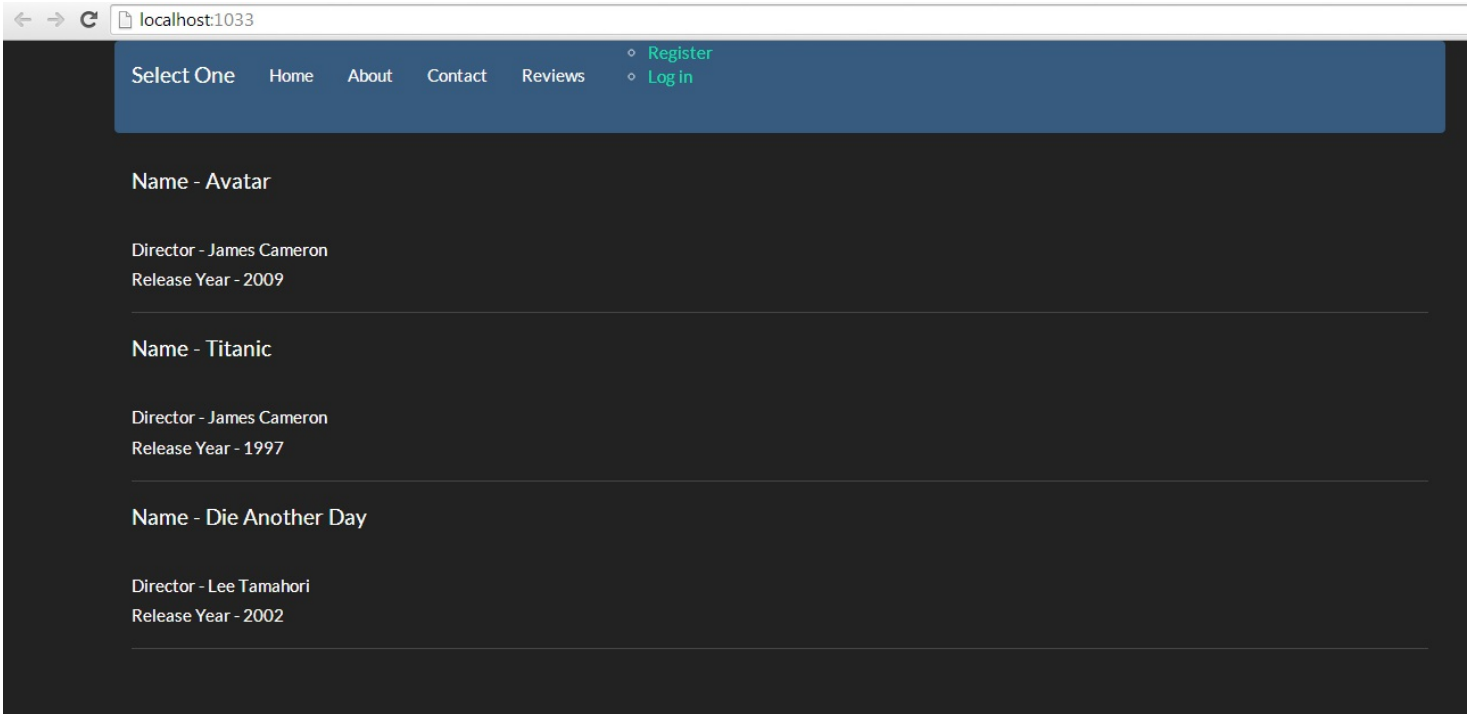


Now, let me change the view a bit and make it more descriptive.

```
namespace MovieReview.Models
{
    public class MovieDb :DbContext
    {
        public MovieDb()
            : base("name=DefaultConnection")
        {

        }
        public DbSet<Movie> Movies { get; set; }
        public DbSet<MovieReviews> MovieReviews { get; set; }
    }
}
```

Now, when I refresh the page, it will produce me the below result.



Database Migrations:-

Now, the next thing is database migartions. In this section, we'll discuss how to influence EF so that I can provide my custom settings saying like where to create the database, what could be the database name, table name etc. So, we can obviously configure data

Context with the specific connection string.

In order to achieve the same, we need to call the base class constructor and then we can provide the explicit connection string with default settings as shown below. But, it's obviously a good practice not to hard code things like connection string, because this will create dependency always to compile the solution, just for changing the database name. So, it's a good idea to specify the same in **web.config** file and fetch the same from here itself.

```
<connectionStrings>
  <add name="DefaultConnection" connectionString="Data Source=.\SQLEXPRESS;Initial Catalog=MoviesReview;Integrated Security=SSPI;AttachDBFilename=|DataDirectory|\MoviesReview.mdf" providerName="System.Data.SqlClient" />
 </connectionStrings>
```

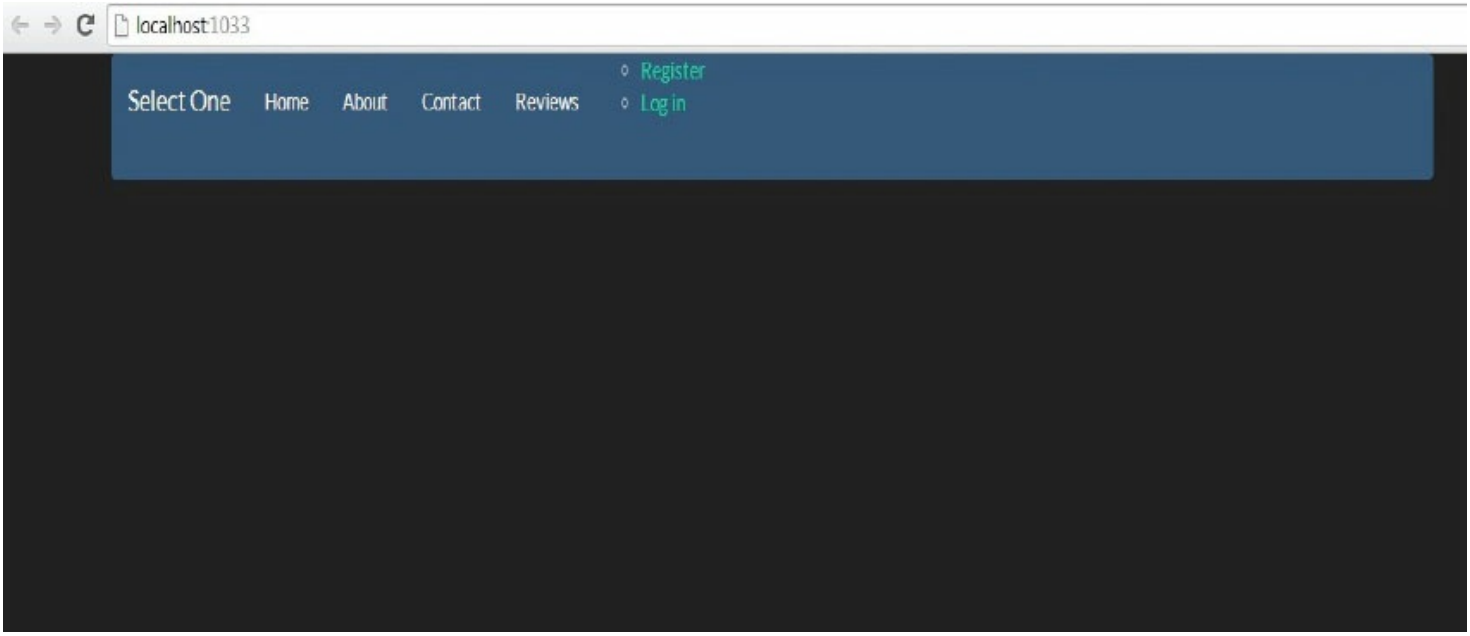
In the above snippet, I have just modified the default value presented to me by visual studio by my custom value. And now, to use the same I just need to specify the connection name in the constructor as shown below.

```
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Web;
```

```
namespace MovieReview.Models
{
    public class MovieDb :DbContext
    {
        public MovieDb()
            : base("name=DefaultConnection")
        {

        }
        public DbSet<Movie> Movies { get; set; }
        public DbSet<MovieReviews> MovieReviews { get; set; }
    }
}
```

Now, when I build the app and refresh the page, all the values will be wiped out as we are now pointing to the new database.



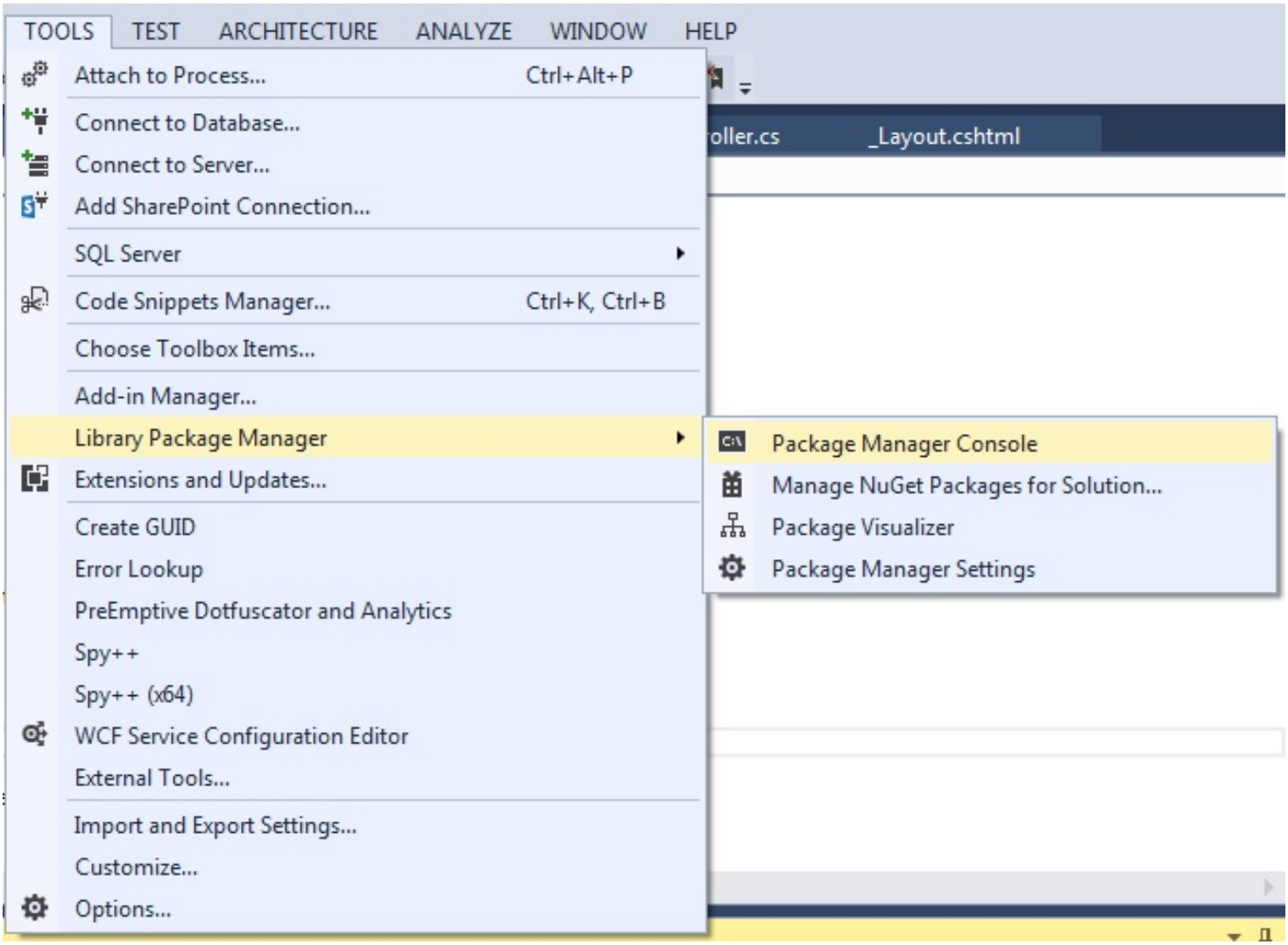
Now, in order to prove the theory that new database has been created with my custom settings, just click on the show all files symbol in the solution explorer and then expand the "***App_Data***" folder, it will list me my custom database as well.



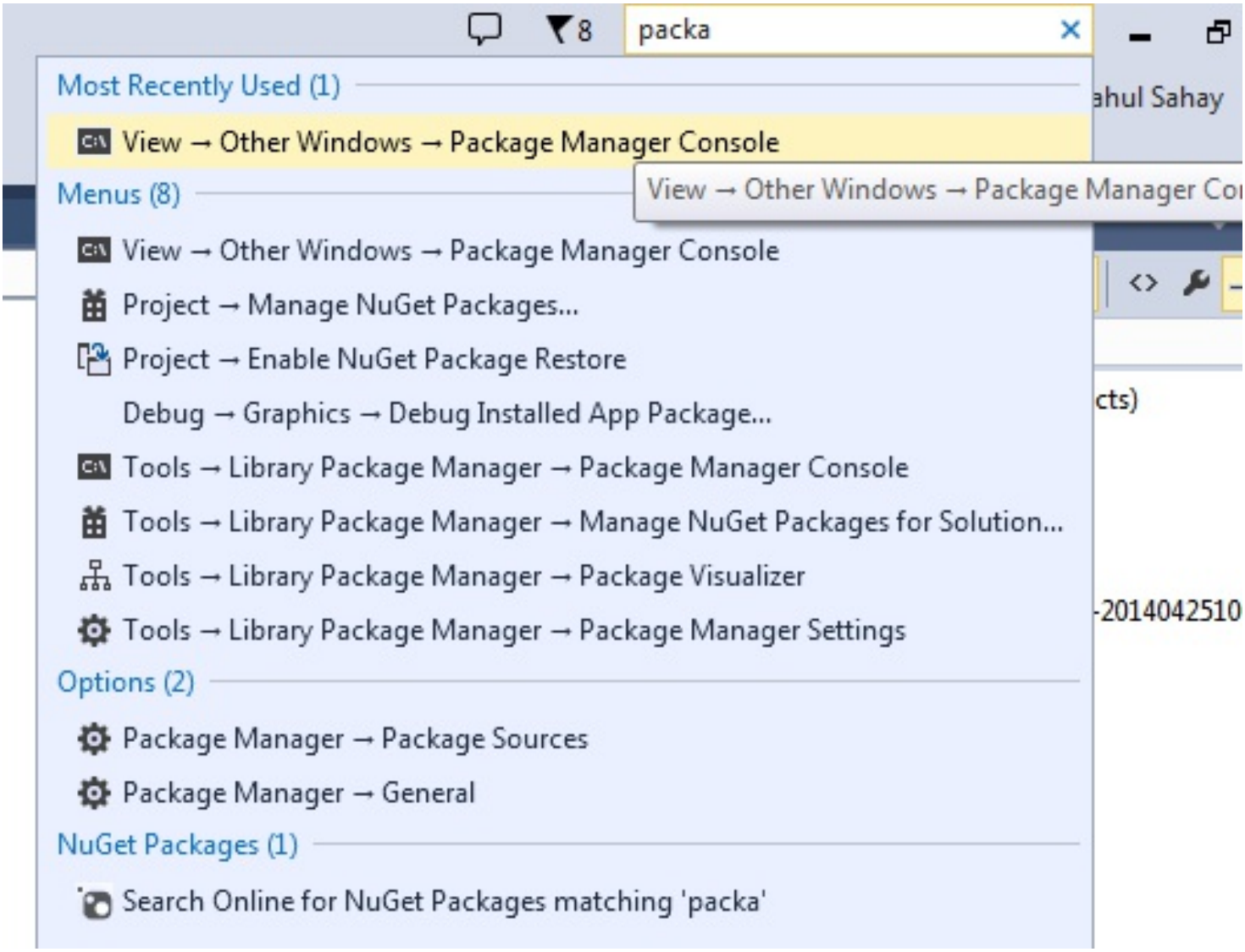
1st one is the default one. 2nd one is the one which I created just now. We can also put some default values same way what we did earlier. But, that's ok we know how to play around. So, moving ahead now think on the point that how could we influence the schema and how could we ensure that it will present some sample data to test my app. And to answer both of these questions, we need to make use of **EF-Migrations**.

Migartions are basically one of the features of **EF** that enable you to configure database schema, seed your database and then track the changes that you are making in the entity classes which you have created earlier. So,

basically this feature enable you to keep the database in sync with the changes which you are making during the course of development. So, to get started with Migartions, go to "***Package Manager Console***". In order to open the same, either you can go to

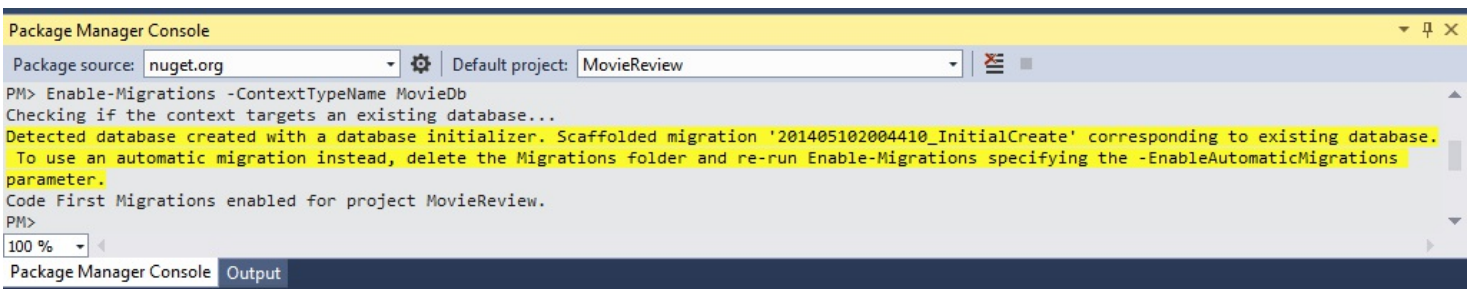


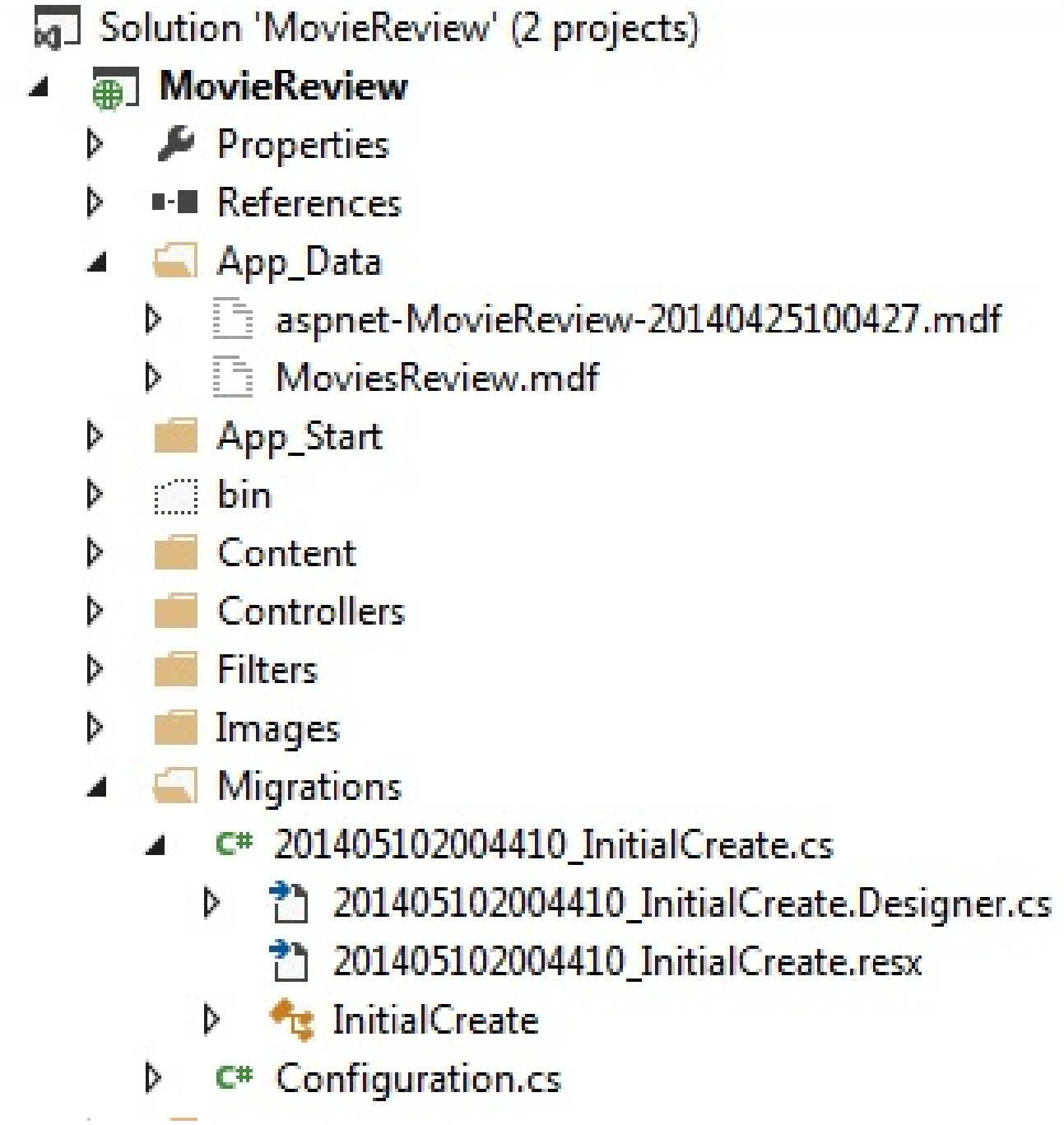
Or directly type in the Quick Launch Window, this will open the console for you. I am quite habituated with this approach. Now, package manager console will open as another window in the Visual Studio itself.



It's basically a NuGet command line. So, now in there I need to enable the migrations. Now, with below mentioned command I also need to specify the "***contextTypeName***". So, basically "***dbContext***" name in which I want to enable migrations.

"Enable-Migrations -ContextTypeName MovieDb". Now, upon running this poweshell command, it will add a new folder to my solution and the new folder is called "***Migrations***".





Now, let's dig inside the file. So, when you open the **Configuration** class, you will come to know that configuration class is all about controlling Code First

Migrations. Settings like performance, when should it run etc.

```
namespace MovieReview.Migrations
{
    using System;
    using System.Data.Entity;
    using System.Data.Entity.Migrations;
    using System.Linq;

    internal sealed class Configuration : DbMigrationsConfiguration<MovieRevi
    {
        public Configuration()
        {
            AutomaticMigrationsEnabled = false;
        }
```

```
            //    context.People.AddOrUpdate(
            //      p => p.FullName,
            //      new Person { FullName = "Andrew Peters" },
            //      new Person { FullName = "Brice Lambson" },
            //      new Person { FullName = "Rowan Miller" }
            //    );
            //
        }
    }
}
```

Now, in the above code snippet, if you see, there is one setting which sets "***AutomaticMigrationsEnabled"*** By default, it is set to false which means that **EF** won't make any changes unless you explicitly tell the **EF** to do so. But, during development, it is good idea to set it true. Because you would be changing different entities or adding few

other entities for your testing purpose. Another nice thing about this configuration file is the seed method. This is the place wherein you instruct EF to push some initial data. So, everytime, when it's going to update the database, it will invoke the seed method and hence populate the tables.

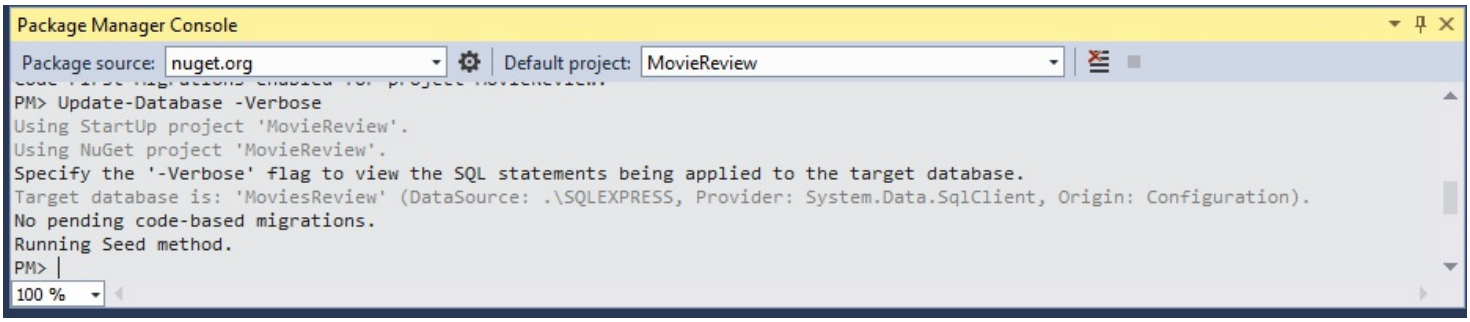
```
protected override void Seed(MovieReview.Models.MovieDb context)
    {
      // This method will be called after migrating to the latest version.

      context.Movies.AddOrUpdate(r => r.MovieName,
        new Movie { MovieName = "Avatar", DirectorName = "James Cameron", ReleaseYear = "2009" },
        new Movie { MovieName = "Titanic", DirectorName = "James Cameron", ReleaseYear = "1997" },
        new Movie { MovieName = "Die Another Day", DirectorName = "Lee Tamahori", ReleaseYear =
"2002" },
        new Movie
        {
          MovieName = "Godzilla",
          DirectorName = "Gareth Edwards",
          ReleaseYear = "2014",
          reviews = new List<MovieReviews>{
            new MovieReviews{ReviewerRating=5,ReviewerComments="Excellent",ReviewerName="Rahul
Sahay"}
          }
        });
    }
```

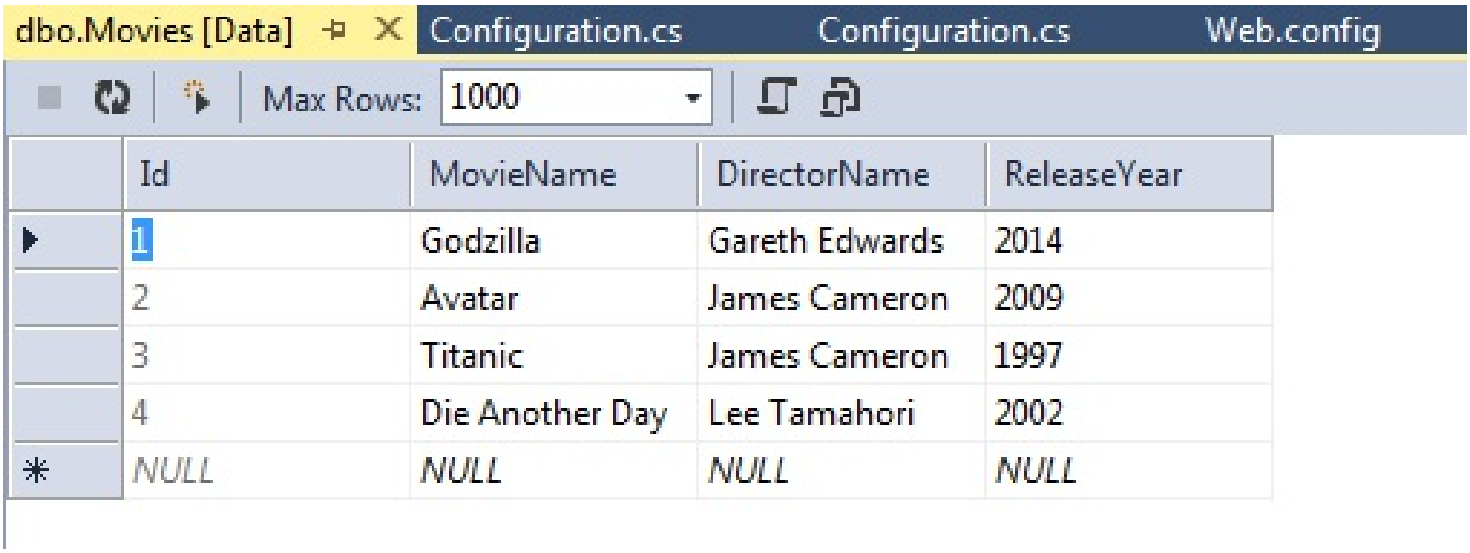
Now, let me explain the above piece of code. So, now this is going to the MovieDb context and saying add or update the following Movies. So, it means before adding or updating the same in the database, it basically checks based on the MovieName, whether the same movie exists or not, if yes, then it will update or it will just add a new record to the database. However, for the last record it will push review in the database as well. This is one of the best ways to push some test data into the db to test your app. Now, inorder to push these values or update these values in the database, again we need to use NuGet command as shown below:-

"***Update-Database -Verbose***".

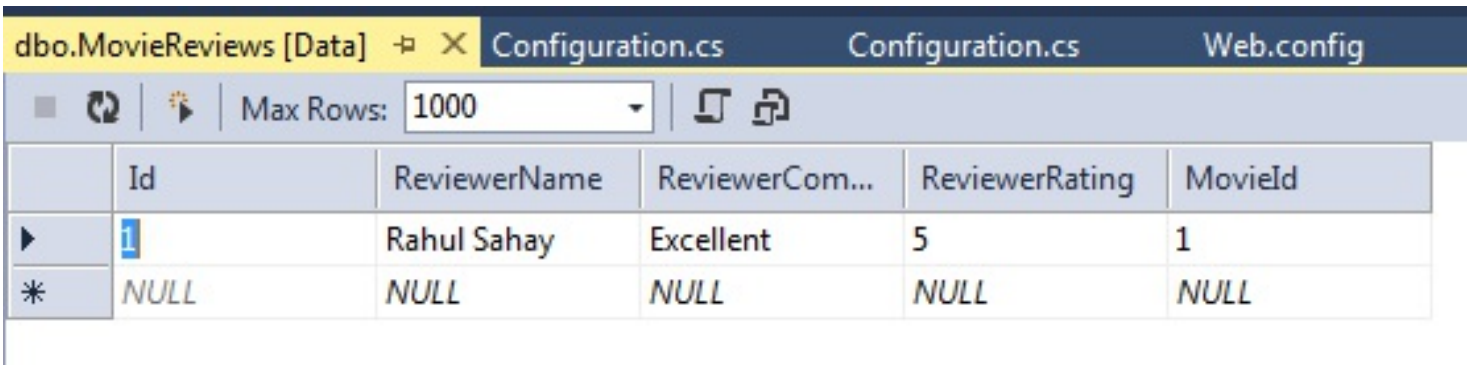
However, for the 1st update, Verbose has nothing much to add, as we haven't done any changes yet in the entities. But, Verbose is going to be extremely useful later in the course. But, for now just run the command.



Once, the update command is completed, we should be able to see the change in the database. Let's go to the server explorer and open the table to view the same.



Id	MovieName	DirectorName	ReleaseYear
1	Godzilla	Gareth Edwards	2014
2	Avatar	James Cameron	2009
3	Titanic	James Cameron	1997
4	Die Another Day	Lee Tamahori	2002
NULL	NULL	NULL	NULL



Id	ReviewerName	ReviewerCom...	ReviewerRating	MovieId
1	Rahul Sahay	Excellent	5	1
NULL	NULL	NULL	NULL	NULL

So, my both the tables have been successfully updated. Now, let's run the app, and see whether correct values are populating or not.

Name - Godzilla

Director - Gareth Edwards

Release Year - 2014

Name - Avatar

Director - James Cameron

Release Year - 2009

Name - Titanic

Director - James Cameron

Release Year - 1997

Name - Die Another Day

Director - Lee Tamahori

Release Year - 2002

So, it looks like app is running fine.

LINQ:-

Now, we have setup our database successfully. Now it's time to move ahead and use **LINQ** to query against the database, and also to save the records in the database. I am assuming that you guys are having some sort of familiarity with **LINQ(Language Integrated Query)**. However, in order to do deep dive with LINQ, please refer MSDN document(***http://msdn.microsoft.com/en-us/library/bb397933.aspx***) for the same. In this segment I am just going to use

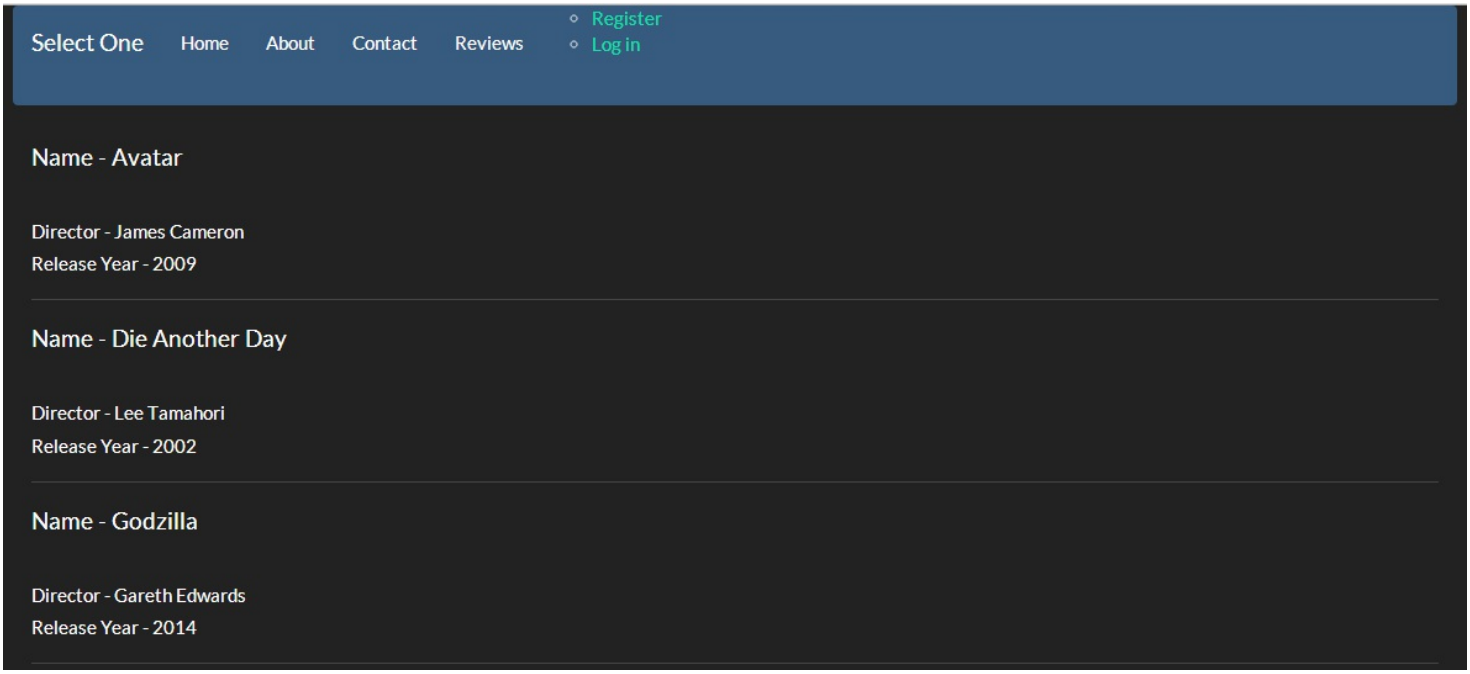
those queries only which are useful in terms of the current application. However, few basic things let me brief you about LINQ queries, say these LINQ queries are strongly typed. So, it means if you change the structure of the class, then these LINQ queries will also get impacted. Remember, the "***MovieReview***" controller code which I have commented because I have changed the entities definition, hence the query broke. There are many targeted things which you could easily do with LINQ queries like ***projecting, filtering, joining etc***. There are specific keywords in C# which let you do the same. We'll see some of these in action later in this segment. Now, let's go ahead

and use some of the queries in our example.

```
public ActionResult Index()
        {

            var model = from m in _movieDb.Movies
                        orderby m.MovieName
                        select m;
            return View(model);
        }
```

In the above section, I am querying against the Movies table using the db context wherein am listing the movies based on the movies name in ascending order. Now, if I build and refresh the page, it will produce me the below result.



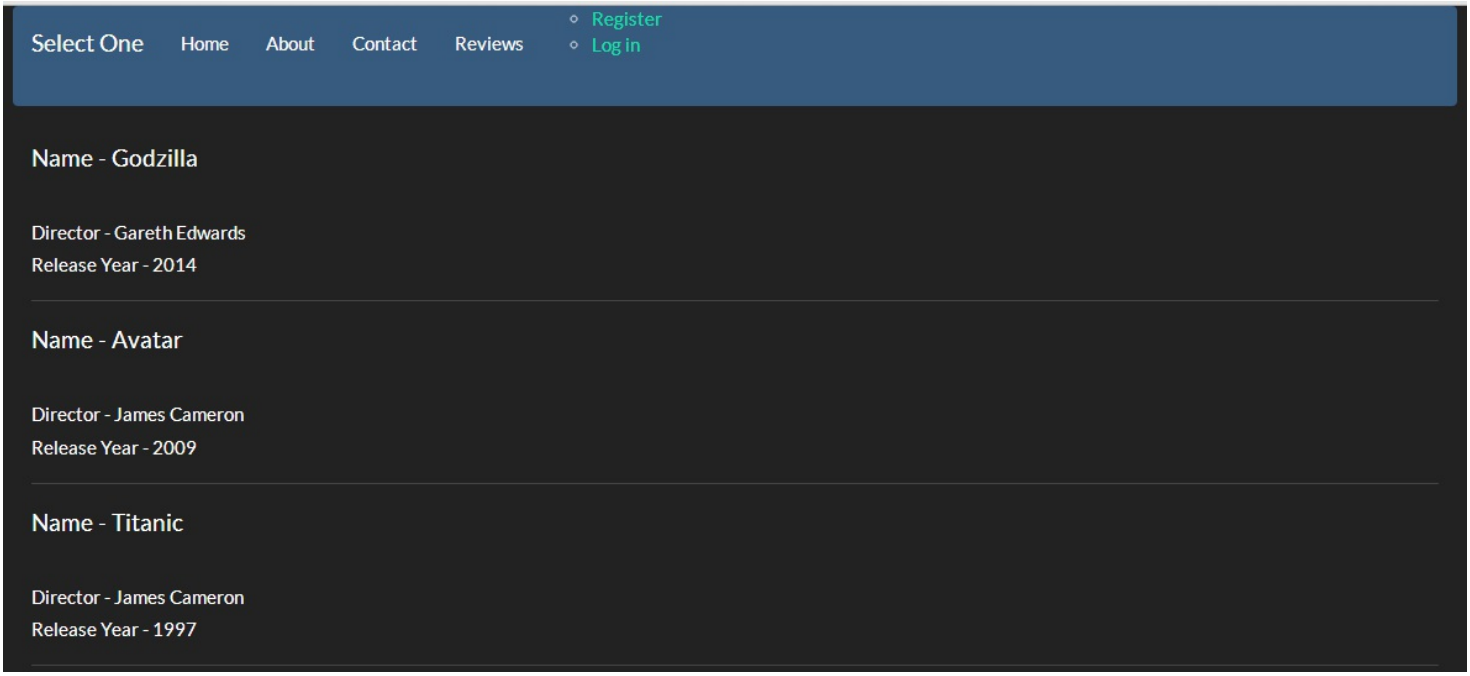
This was one of the basic queries which we usually write. Now, let me go ahead and try some other thing, how about listing the movies based on the reviews, means any movie which has received higher no of reviews is going to appear 1st in the list. In order to achieve the same, I just need to modify the query a bit.

```
public ActionResult Index()
        {

            var model = from m in _movieDb.Movies
                        orderby m.MovieName
                        select m;
            return View(model);
        }

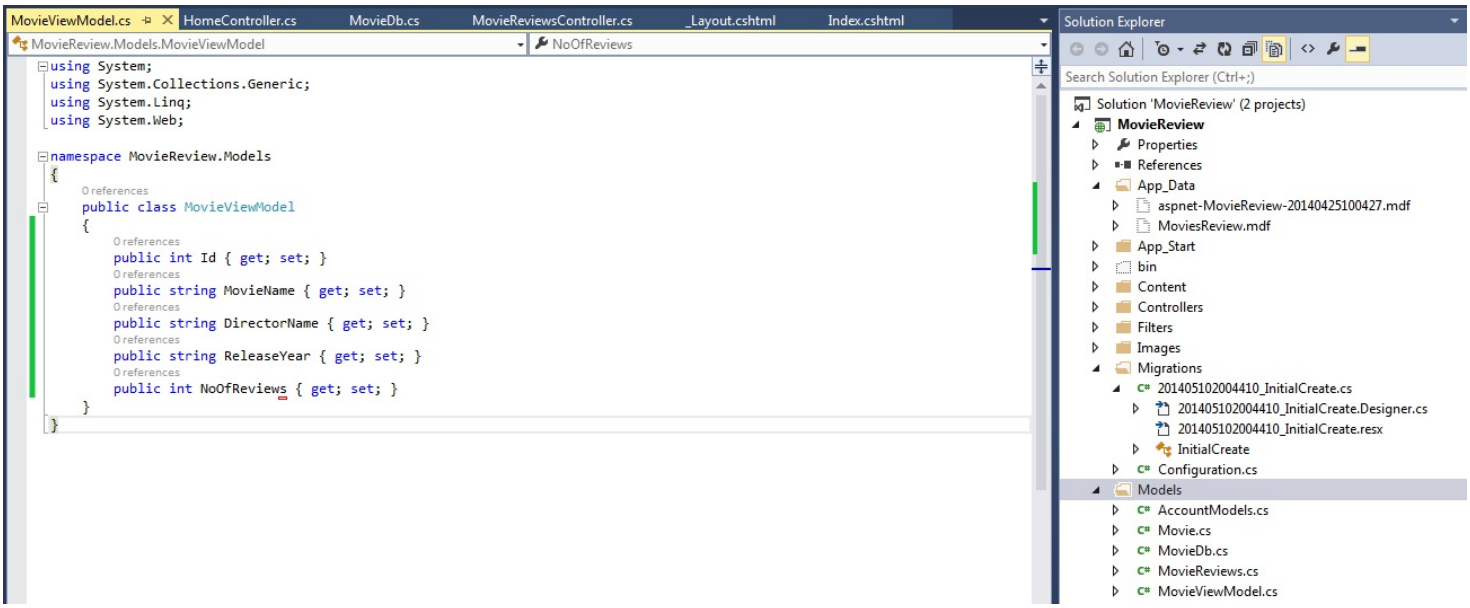
            orderby m.reviews.Count() descending
            select m;
    return View(model);
}
```

Now, with the above piece of code in place, I should be getting Godzilla 1st.



So, here what this LINQ query is doing; taking the LINQ query and transalting the same into a query which is understood by **Sql Server** and then bringing back the result but again before producing the result, it just transforms the result into object which I could use in C# code. Now, let me try one more thing, how about producing a result which is also showing the no of reviews associated with movie.

In order to achieve the same, I need to make use of **ViewModel**, **ViewModel** is again a data-storage concept for the views, this is basically used to simplify our query, or to make the query strongly typed against one. So, I need to create one class and then need to put the properties which I am intrested in.



Now, I am going to use the above class in the model shown below:-

```
public ActionResult Index()
    {

        var model = from m in _movieDb.Movies
                    orderby m.reviews.Count() descending
                    select new MovieViewModel
                    {
                        Id = m.Id,
                        MovieName = m.MovieName,
                        DirectorName = m.DirectorName,
                        ReleaseYear = m.ReleaseYear,
                        NoOfReviews = m.reviews.Count()
                    };
        return View(model);
    }
```

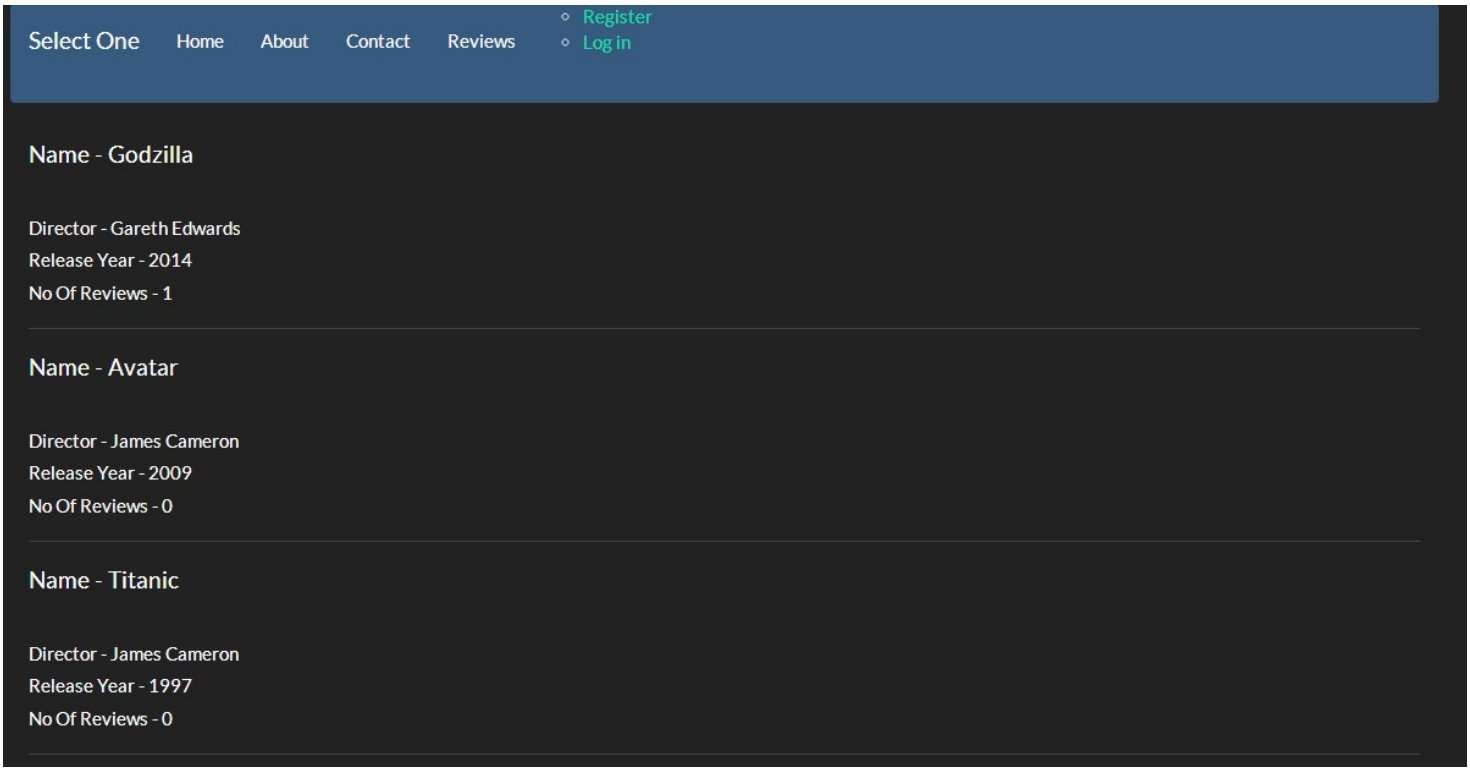
So, viewModel is basically going to carry the information which the view needs to display. Now, I also need to modify my Index View. So, now my

view looks like

```
@model IEnumerable<MovieReview.Models.MovieViewModel>
@{
    ViewBag.Title = "Home Page";
}

@foreach (var item in Model)
{
    <div>
        <h4>Name - @item.MovieName</h4><br />
        <h5>Director - @item.DirectorName</h5>
        <h5>Release Year - @item.ReleaseYear</h5>
        <h5>No Of Reviews - @item.NoOfReviews</h5>
        <hr />
    </div>
}
```

Now, when I build and refresh the page, it will produce me the below output.



Search Query:-

Now, the next feature which I want to discuss is the filtering facility in the application. So, in order to achieve the filtering I just need to add one parameter in the Index method say "***SearchQuery***". So, I'll allow users to provide one filtering criteria to search the result. By

default it would be null.

```
public ActionResult Index(string SearchQuery = null)
    {

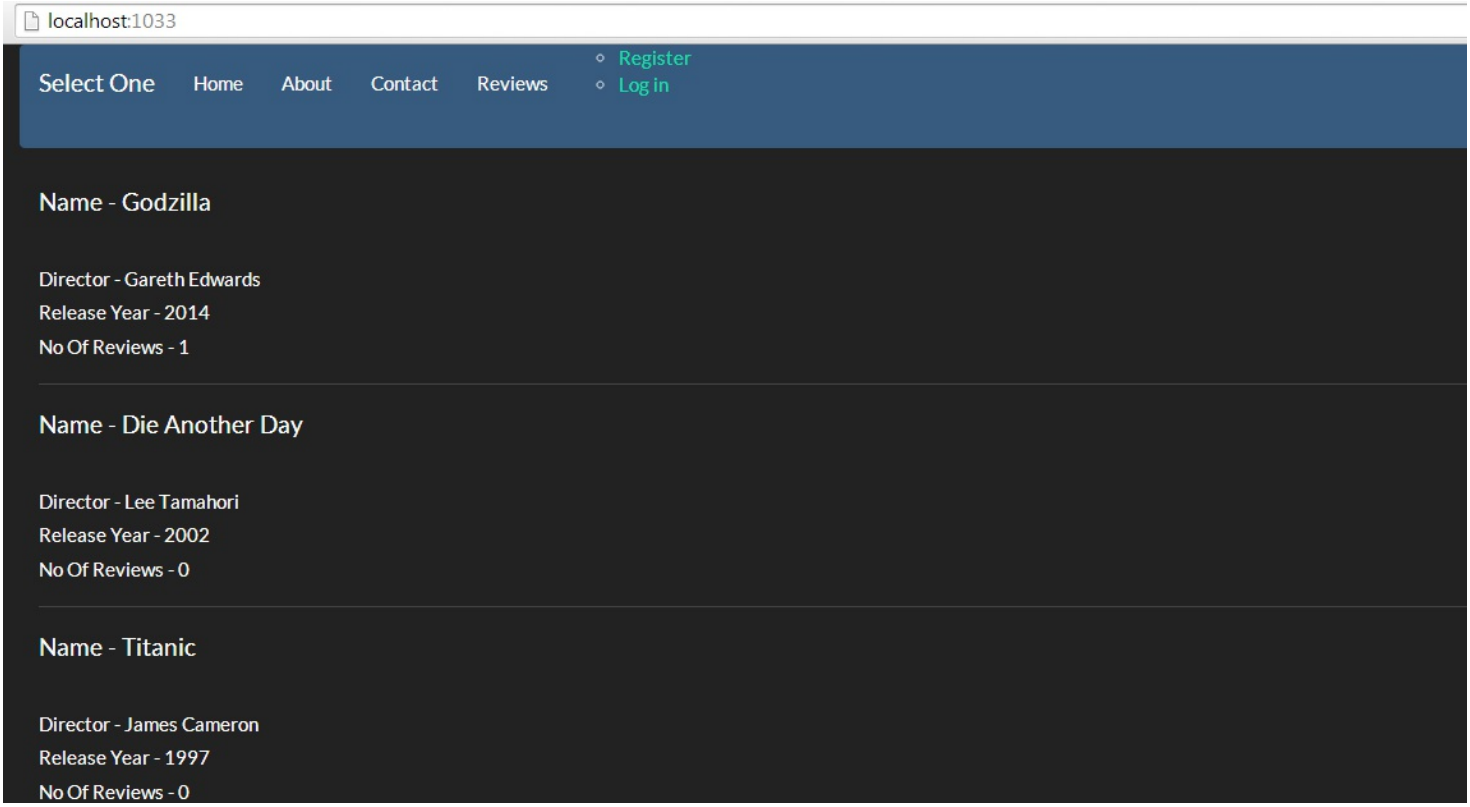
        var model = _movieDb.Movies
                .OrderByDescending(m => m.reviews.Count())

        .Where(m => SearchQuery == null || m.MovieName.StartsWith(SearchQuery))
        .Take(3)
        .Select(m => new MovieViewModel
        {
           Id = m.Id,
           MovieName = m.MovieName,
           DirectorName = m.DirectorName,
           ReleaseYear = m.ReleaseYear,
           NoOfReviews = m.reviews.Count()
        });
  return View(model);
}
```

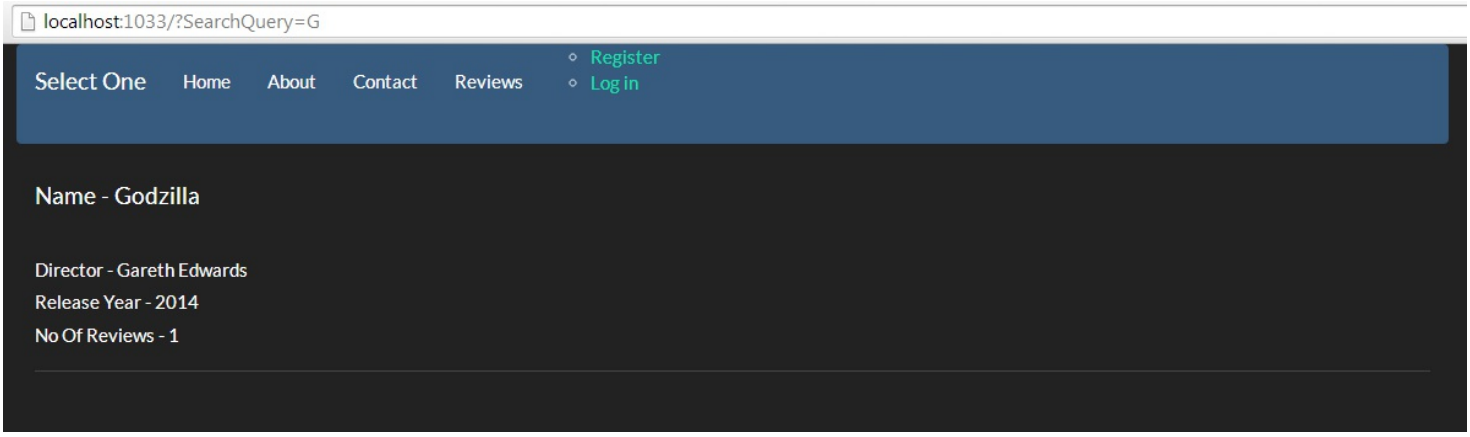
Now, what this above query is doing? If SearchQuery is null, it will return all the movies. However, if SearchQuery matches some of the text, then it will

return the desired result. You also might have noticed, that I changed the LINQ query completely from the above section, just to make sure that I can use pagination.

Because later in coming chapter, we'll discuss the same. However, this kind of query is also referred as using "***Extension Methods***". Now, in order to verify whether the same is working correctly or not, just build the app and refresh the page.



So, now it's listing only 3 records. Now, to test the search criteria, let's pass in the value as query string and then search.



So, in the above screen shot I have

passed in "***G***" as search parameter which worked correctly and fetched the appropriate result. That's nice, but from the end user's perspective but, let's give user; ability to enter the **Search Text** in the form itself. So, for that we need to do little bit of modification in the view itself.

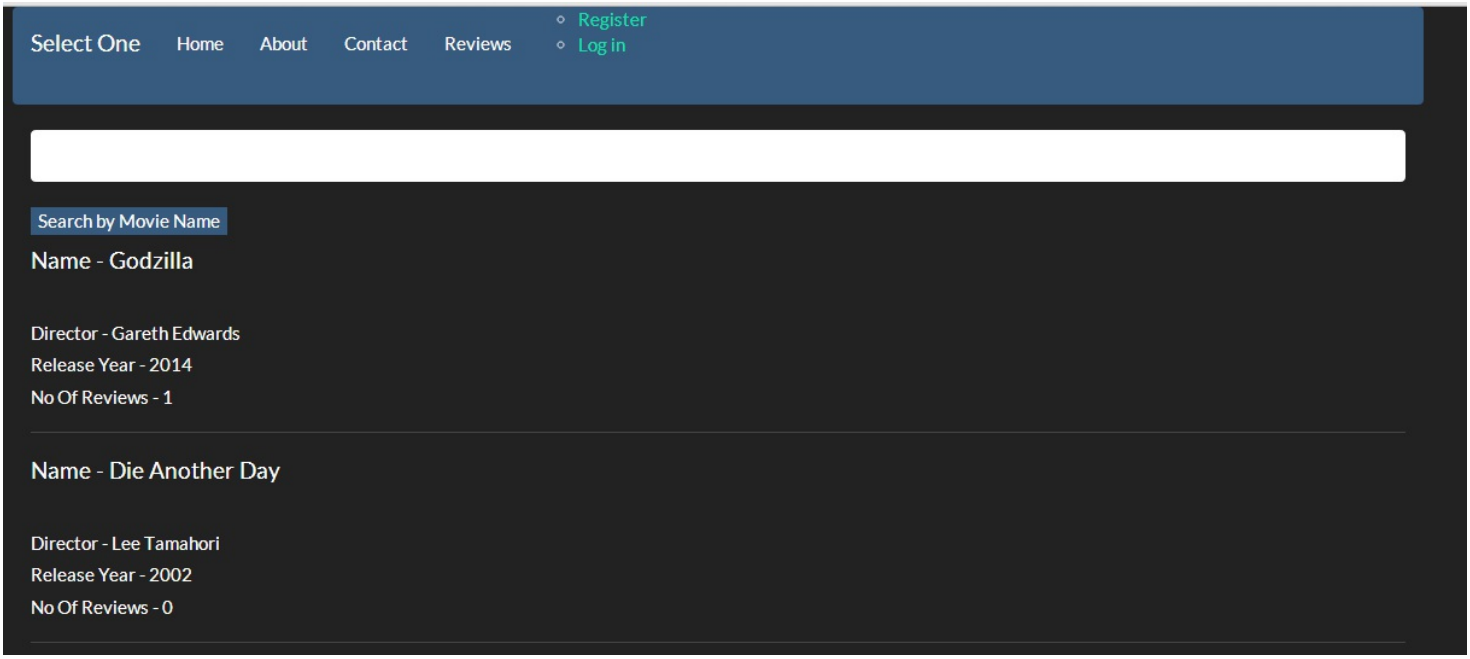
```
@model IEnumerable<MovieReview.Models.MovieViewModel>
@{
    ViewBag.Title = "Home Page";
}

<div>
    <form method="get">
        <input type="search" class="form-control has-success" name="SearchQuery">
        <br />
        <input type="submit" class="btn-primary" value="Search by Movie Name" />
    </form>
</div>

@foreach (var item in Model)
{
    <div>
        <h4>Name - @item.MovieName</h4><br />
        <h5>Director - @item.DirectorName</h5>
        <h5>Release Year - @item.ReleaseYear</h5>
        <h5>No Of Reviews - @item.NoOfReviews</h5>
        <hr />
    </div>
}
```

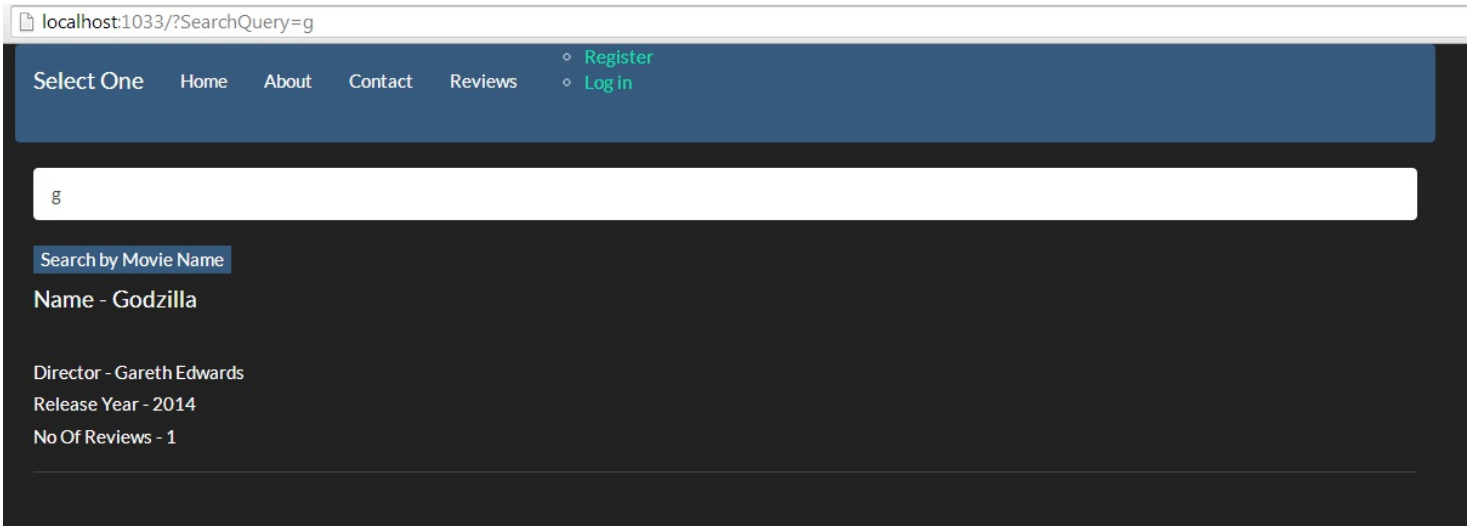
Now, in the above case, if you have noticed that I have introduced one div section where in I have used one form control. Couple of important point to

note here, is like form method should be “get”, so that it can go ahead and call the index method and then name which is equal to “SearchQuery”, this should be the same as I have used in the MVC Index method, then only it can identify the query string parameter. Now, when I refresh the page, it will produce me the below UI.



Now, when I type something in search

box and do the search query, it should produce me the same result, what it produced earlier like shown below.



One point to note here, because of form method “get” , the search string appears in the query string. So, when you click on the submit button for a form which issues a get request, it will encapsulate all the input values and put the same in the query string. On, the other hand, if we would have used post method, then

in that case all the values would have been encapsulated within http Message body. One must understand the contrasting difference between get and post operation, before using them. Get operations are basically search operations and Post operations are write operations where in you are writing something to the database.

Summary:-

So, in this chapter, we have started working on real time database by setting it up and running. We have also seen how Entity Framework works in real time. We have seen the contrasting

difference between EF Default settings and custom settings. We have also pushed the values manually and using migrations in the database. For using the same we have seen couple of NuGet commands in action. We have also covered different kind of mature LINQ queries to use against the database. And last but not the least we have enabled search facility for the user to search the movie.

Chapter 5: Data modeling- Part 2

WHAT DO you find in this CHAPTER?

⇒ Introduction

⇒ Organizing Movie

Controller

⇒ Reviews Listing

⇒ Create and Edit Facility of Review

⇒ Validation Annotations

⇒ Summary

Introduction:-

Hi, good to see that you guys have been enjoying the book and reached the 2nd part of data modeling. So, far in this journey we have done quite a significant piece of development and in this chapter,

we will be extending the data model part which we have started in the last chapter. So, without wasting time, let's get back to business.

Organizing Movie Controller:-

Now, in this section what we are going to implement **CRUD** operation. So, in this section we are going to have a movie controller which will do all these things for me. Let's go ahead and create a Movie controller to manage all these things for me. So, the very 1st thing which I am going to do here is replace the link which has been producing

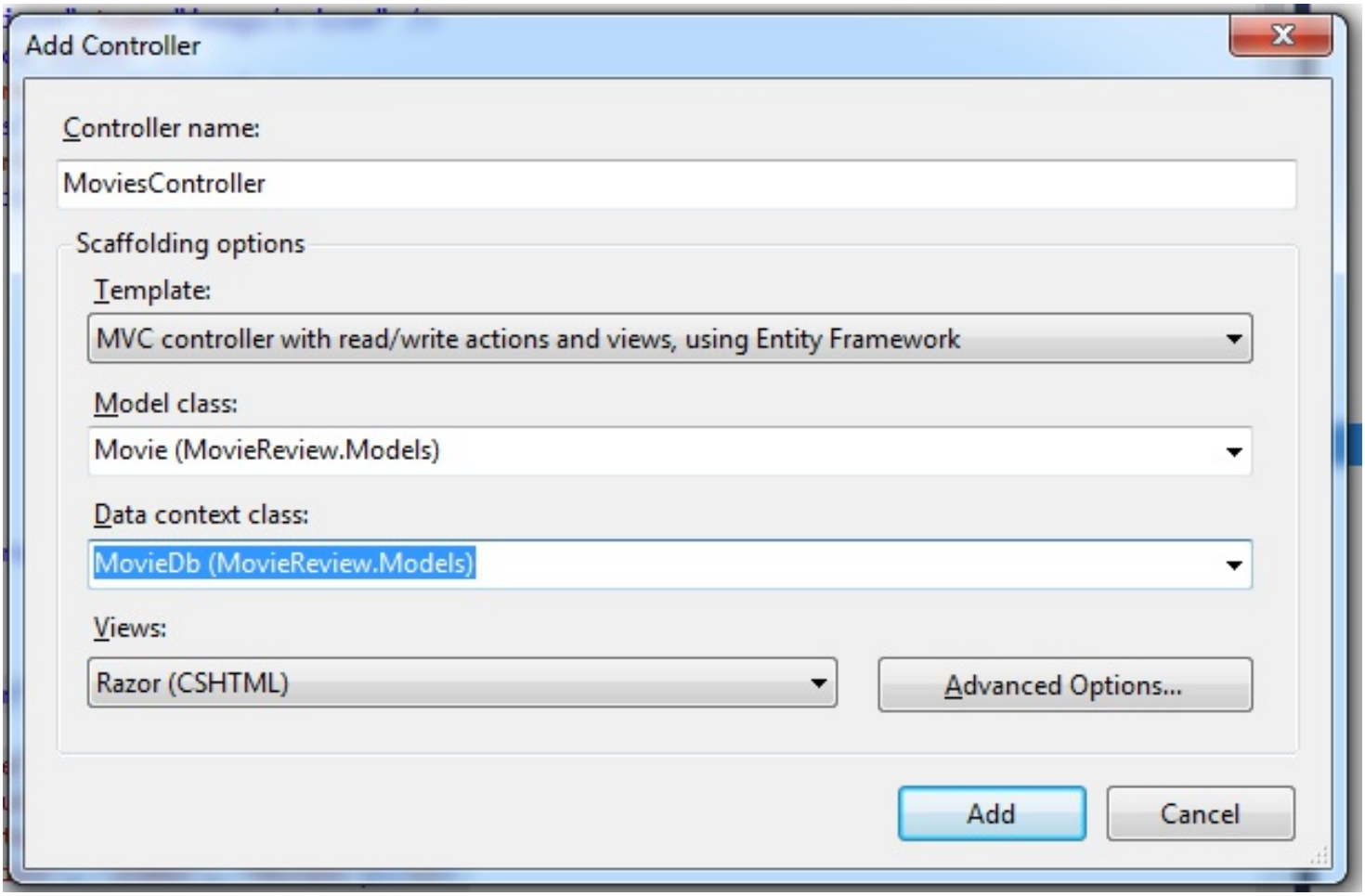
reviews so far, because I have already removed that controller. Now, in here I have introduced new controller as "***MoviesController***".

```
<div class="navbar-collapse navbar">
                <ul class="nav navbar-nav">
                    <li>@Html.ActionLink("Home", "Index", "Home")</li>
                    <li>@Html.ActionLink("About", "About", "Home")</li>
                    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
                    <li>@Html.ActionLink("Movies", "Index", "Movies")</li>
                    <li>@Html.Partial("_LoginPartial")</li>
                </ul>

            </div>
```

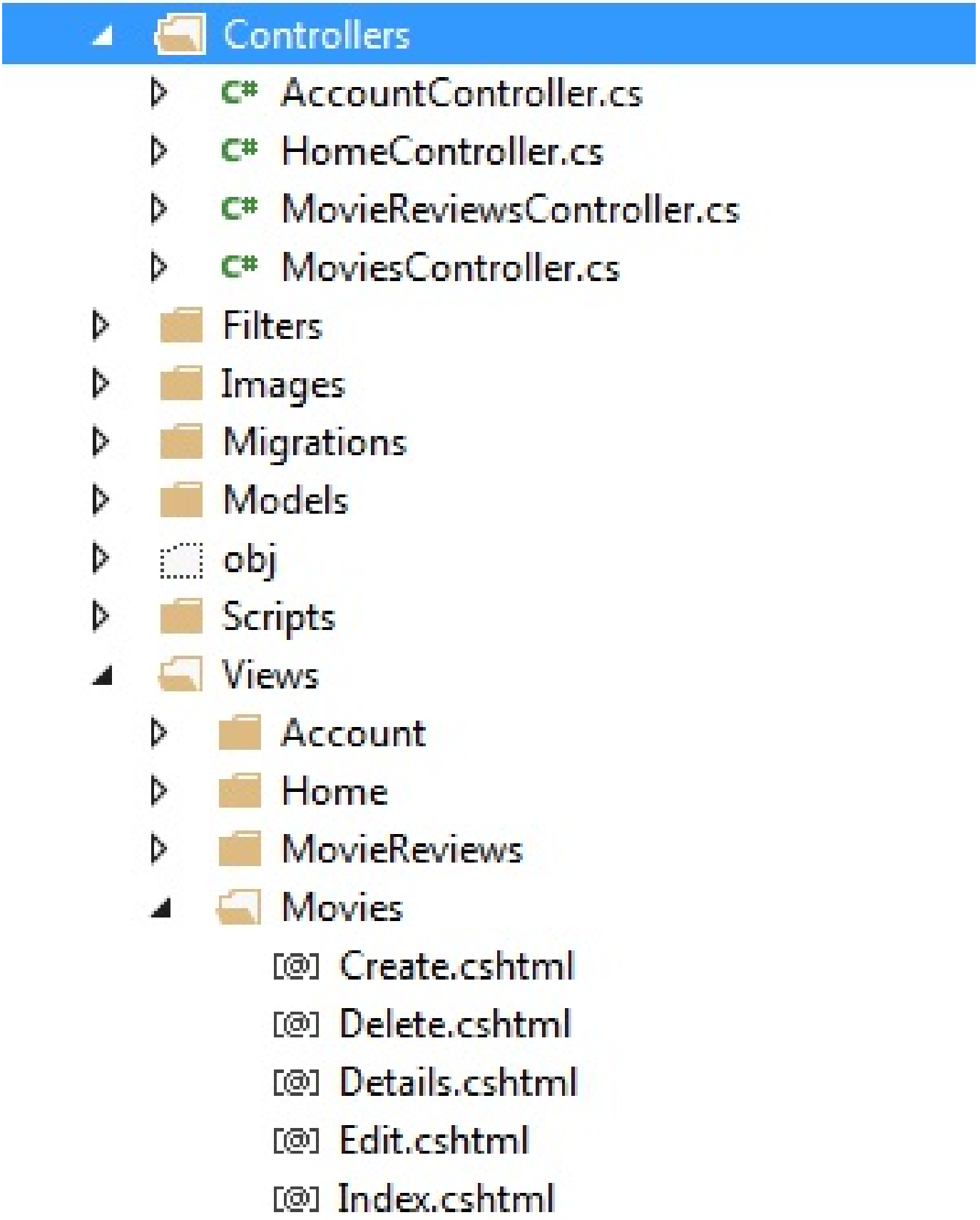
Above snippet is the modified example for the same. However, I am going to create Movies controller as I haven't created that. While creating the controller I am going to select the option MVC controller with "***Read/Write actions with Entity Framework***". I will also provide Model class and data

context class as shown below.



Now, when I click on add, Visual studio will go ahead and all the required components for me which will allow me to save a new record to the database, to query the database and also provide all the required views which I need in the

views folder. And, once this action is completed it will simply open all these files in the editor itself. Below, is the sample snapshot that how it looks in solution explorer.



So, now let's build the application and

try the same. Now, based on my code structure what we have selected while building the Movies Controller, it has given me boiler plate code for all the CRUD Operation. Now, below is the sample snippet which is being generated by visual studio and given to me.

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using MovieReview.Models;

namespace MovieReview.Controllers
{
    public class MoviesController : Controller
    {
        private MovieDb db = new MovieDb();

        //
        // GET: /Movies/
```

```
public ActionResult Index()
{
    return View(db.Movies.ToList());
}

//
// GET: /Movies/Details/5

public ActionResult Details(int id = 0)
{
    Movie movie = db.Movies.Find(id);
    if (movie == null)
    {
        return HttpNotFound();
    }
    return View(movie);
}

//
// GET: /Movies/Create

public ActionResult Create()
{
    return View();
}

//
// POST: /Movies/Create

[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create(Movie movie)
{
    if (ModelState.IsValid)
    {
        db.Movies.Add(movie);
        db.SaveChanges();
        return RedirectToAction("Index");
    }

    return View(movie);
}

//
// GET: /Movies/Edit/5
```

```
public ActionResult Edit(int id = 0)
{
    Movie movie = db.Movies.Find(id);
    if (movie == null)
    {
        return HttpNotFound();
    }
    return View(movie);
}

//
// POST: /Movies/Edit/5

[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit(Movie movie)
{
    if (ModelState.IsValid)
    {
        db.Entry(movie).State = EntityState.Modified;
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    return View(movie);
}

//
// GET: /Movies/Delete/5

public ActionResult Delete(int id = 0)
{
    Movie movie = db.Movies.Find(id);
    if (movie == null)
    {
        return HttpNotFound();
    }
    return View(movie);
}

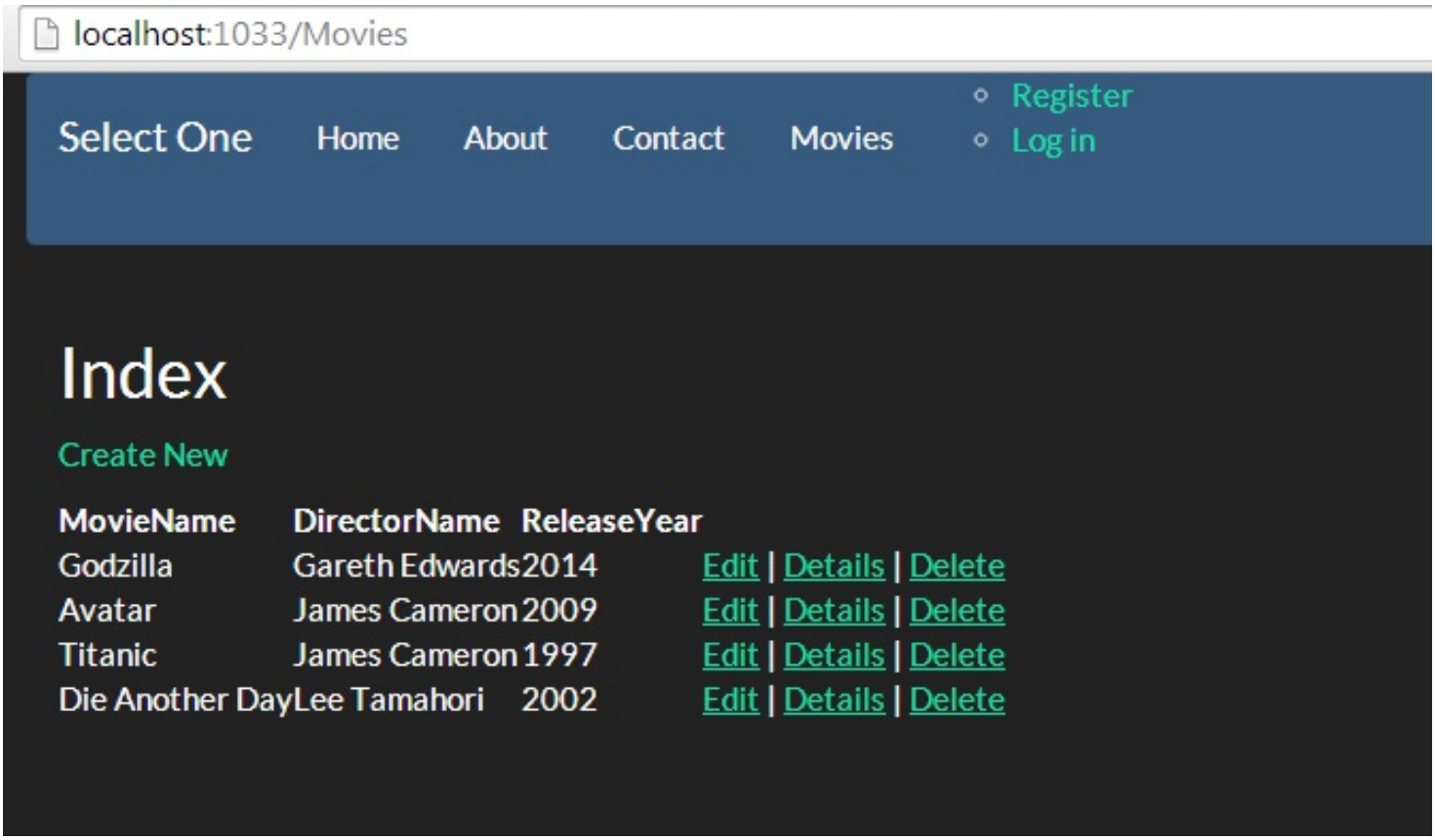
//
// POST: /Movies/Delete/5

[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public ActionResult DeleteConfirmed(int id)
```

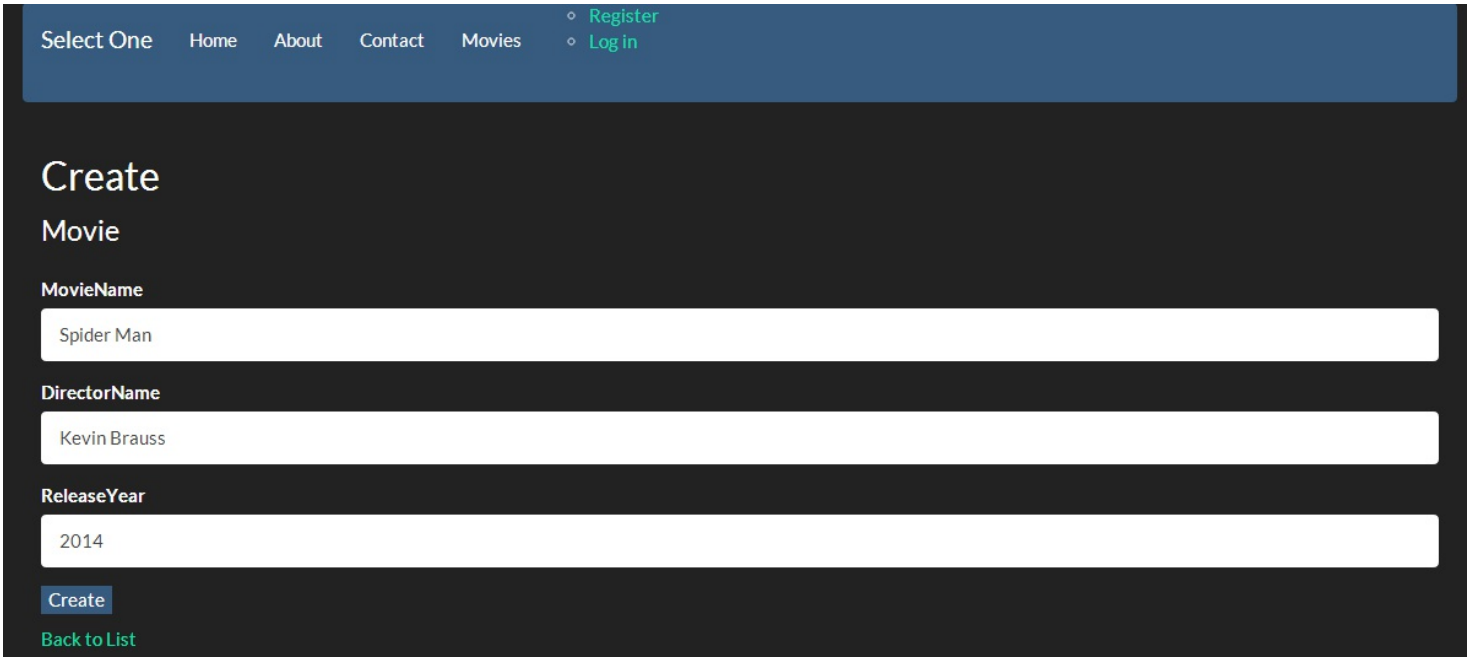
```
        {
            Movie movie = db.Movies.Find(id);
            db.Movies.Remove(movie);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            db.Dispose();
            base.Dispose(disposing);
        }
    }
}
```

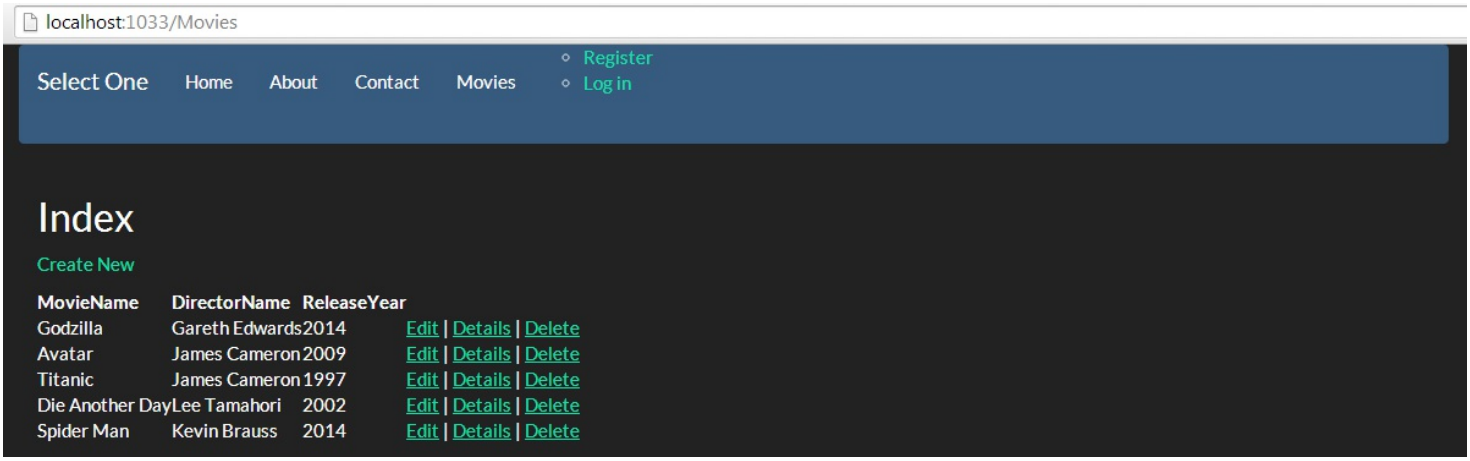
Now, let's see, how the app looks like when I go to the Movies Link.



Now, as you can see there is different link associated for each movie. Each of these links is the part of CRUD operation. Now, with this above view I should be able to create a new movie just by typing the required values in there. So, let's do that.

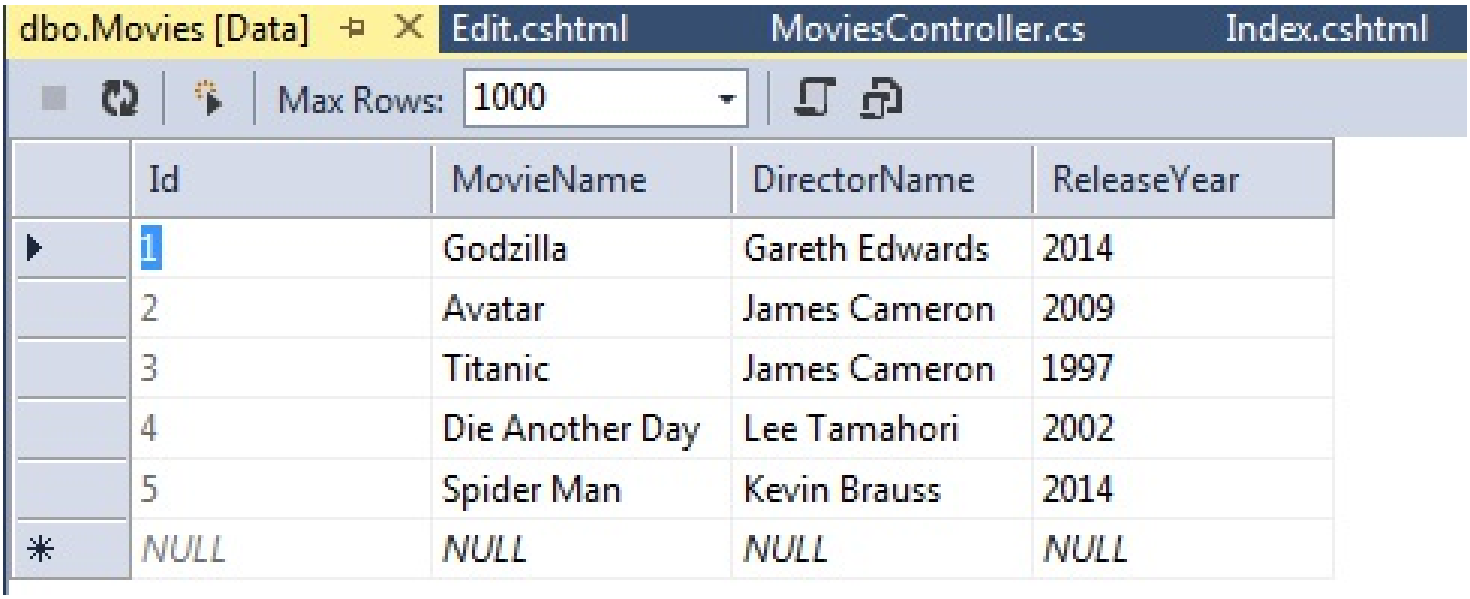


Now, when I click on the link Create, it will create one new movie in the database with the above inputs as shown below.



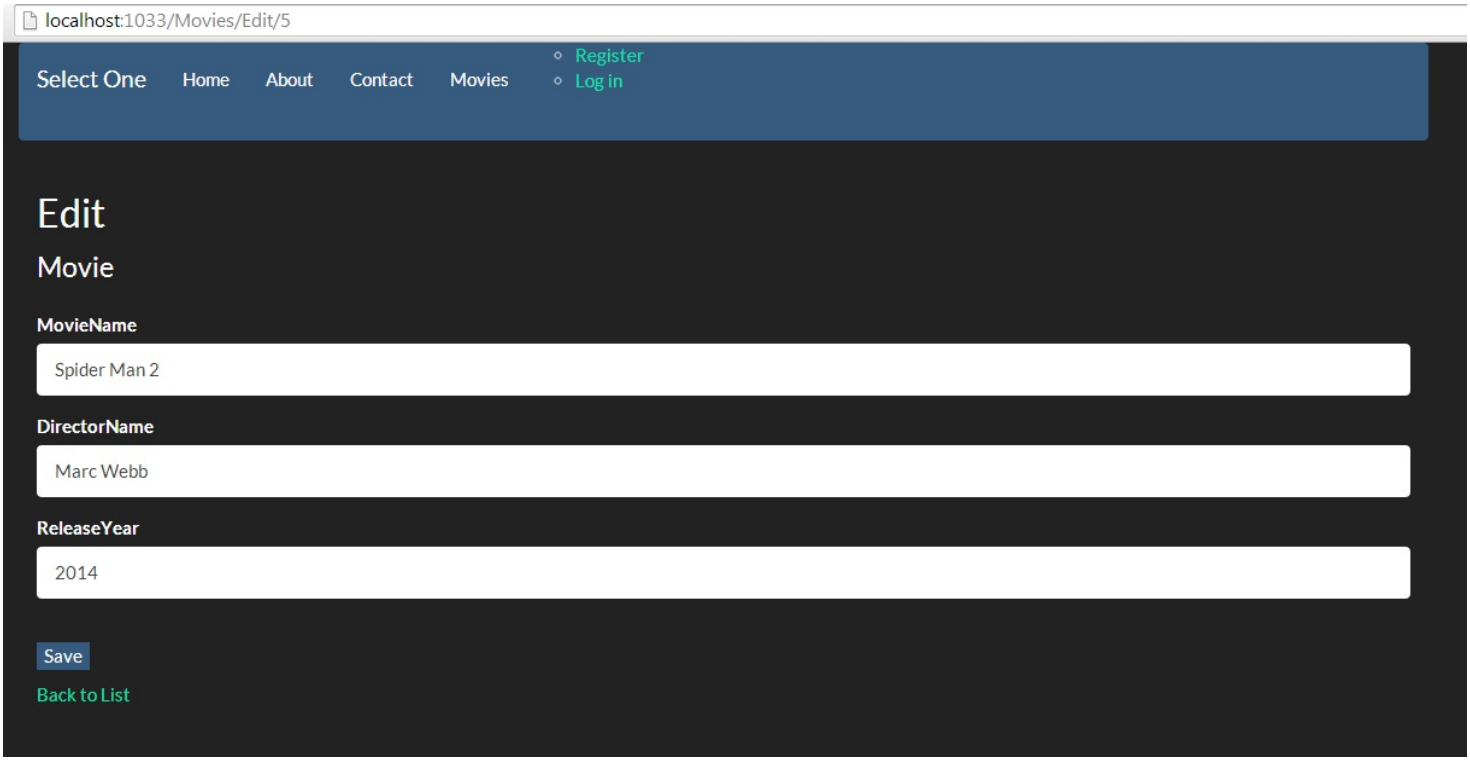
Now, we can also verify the same in the

database also. Let' s do that as well.

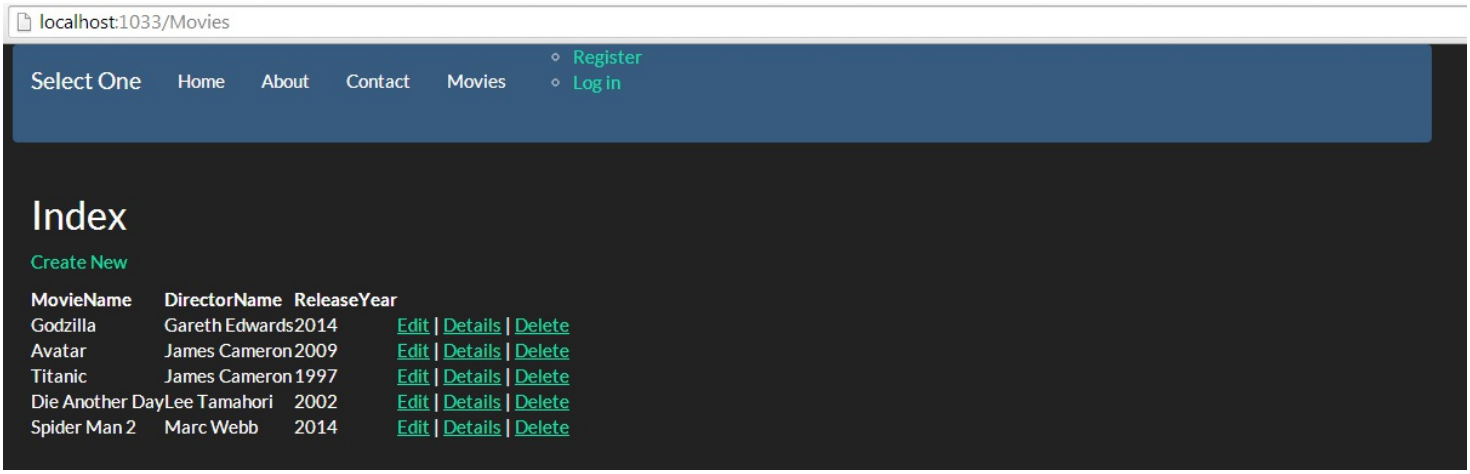


Id	MovieName	DirectorName	ReleaseYear
1	Godzilla	Gareth Edwards	2014
2	Avatar	James Cameron	2009
3	Titanic	James Cameron	1997
4	Die Another Day	Lee Tamahori	2002
5	Spider Man	Kevin Brauss	2014
NULL	*NULL*	*NULL*	*NULL*

So, as you can see that new record has been inserted in the database. Now, let me try other things like checking Edit and Delete functionality as well. Let, me go ahead and change the Title of "***Spider Man***" to "***Spider Man 2***" and see if that works.

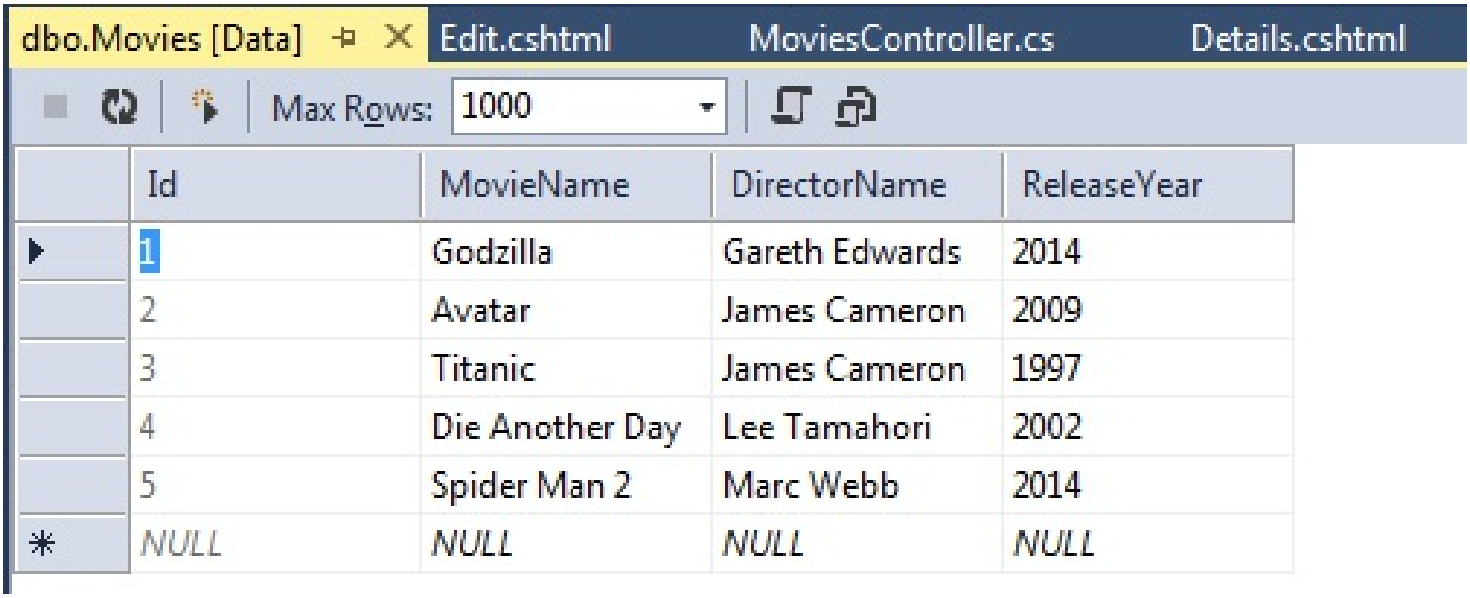


Now, when I save the same it will take the effect.



So, in the above screen shot you can see the change. However, I have changed the

director name as well. Now, when you refresh the database, you will obviously see the change. Let's try that also.

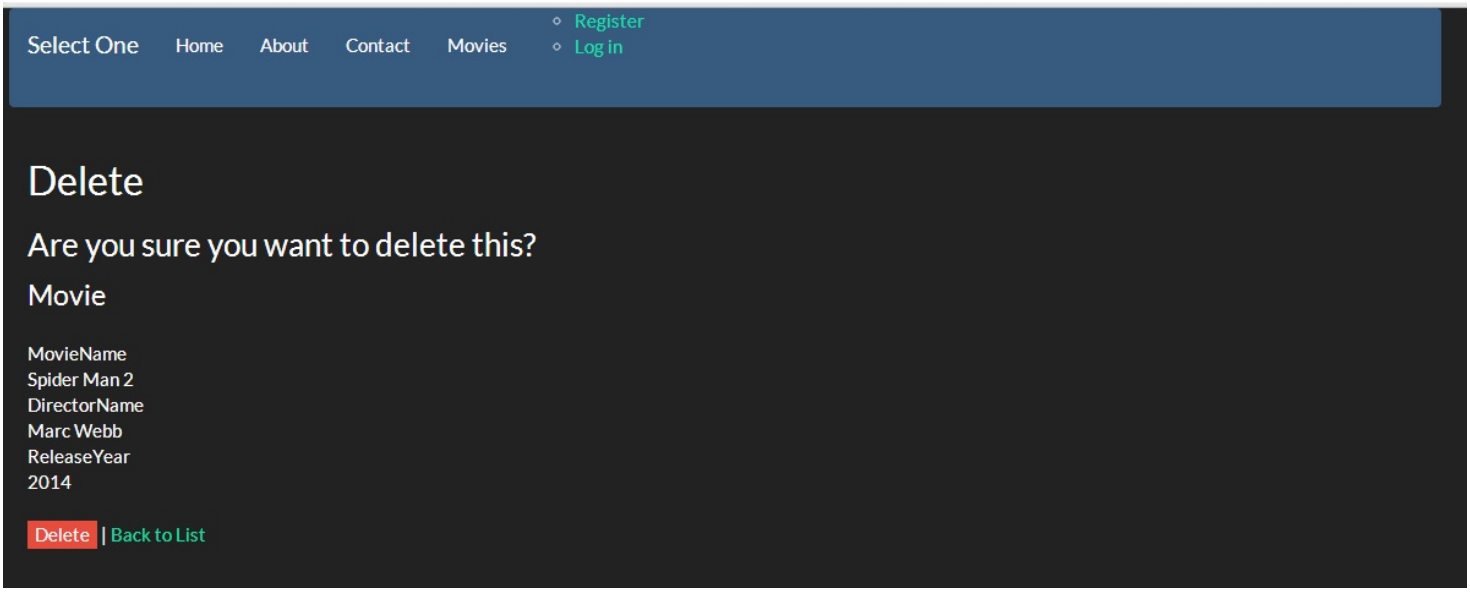

dbo.Movies [Data] | Edit.cshtml | MoviesController.cs | Details.cshtml

Max Rows: 1000

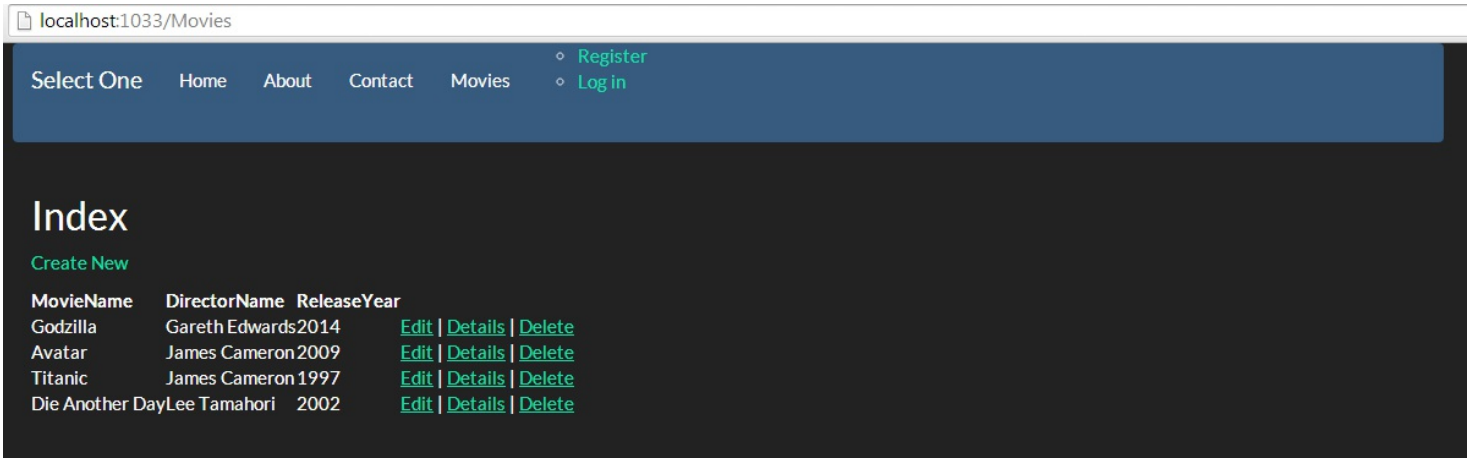
Id	MovieName	DirectorName	ReleaseYear
1	Godzilla	Gareth Edwards	2014
2	Avatar	James Cameron	2009
3	Titanic	James Cameron	1997
4	Die Another Day	Lee Tamahori	2002
5	Spider Man 2	Marc Webb	2014
NULL	NULL	NULL	NULL

So, you can see that the record has been modified. Now, let's try deleting the same and check how it works. So, when you click on delete link against the movie, it will not directly delete the record in the 1st go. 1st it will present the details for you in one simple list and ask for your confirmation. So, if you click

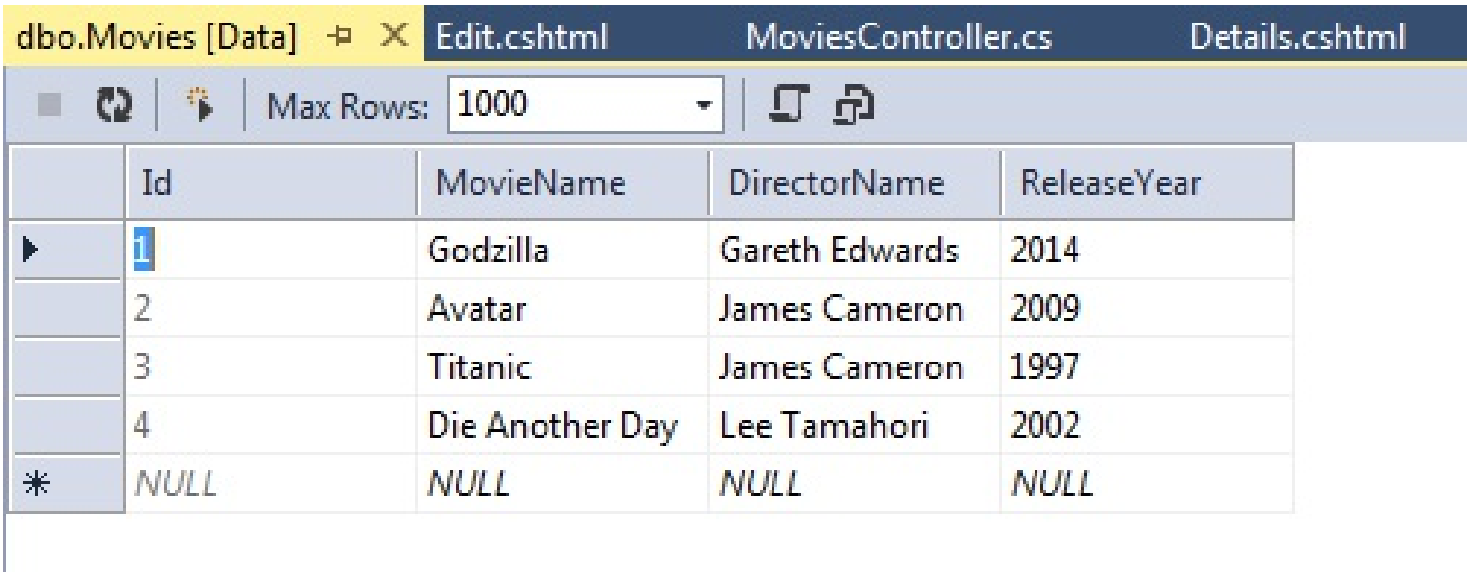
"***Delete***" button on this confirmation screen, then only it will go and delete the record.



So, above screen shot is the confirmation screen where in it's asking you to confirm before deletion. Now, when you click on the delete button, it will simply go and delete the record.



So, as you can see in the above screen shot that "***Spider Man 2***" has been deleted. Let's re-confirm the same from the database side as well.



Id	MovieName	DirectorName	ReleaseYear
1	Godzilla	Gareth Edwards	2014
2	Avatar	James Cameron	2009
3	Titanic	James Cameron	1997
4	Die Another Day	Lee Tamahori	2002
NULL	*NULL*	*NULL*	*NULL*

Now, one point to note here, that there is

a significant difference between the UI which I am presenting here, and which you may encounter during development as Visual Studio will present a basic plain template, which I have changed according to my convenience by using different bootstrap class names, form controls etc. So, in order to get the same feel, either you can download the source code from the github(***https://github.com/rahulsahay19/Review***) or you can download your desired **UI Theme** and style it your way.

Now, this all CRUD operation happened without my explicit code or without any additional code. This all happened because of the scaffolding template

which I have selected while making the "***Movies***" controller. So, thanks to Visual studio for saving time and effort for doing the most obvious thing while writing any app from the scratch. Let's looks at the controller code for a minute and understand what it is doing under the hood.

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Web;
```

```
using System.Web.Mvc;
using MovieReview.Models;

namespace MovieReview.Controllers
{
    public class MoviesController : Controller
    {
        private MovieDb db = new MovieDb();

        //
        // GET: /Movies/

        public ActionResult Index()
        {
            return View(db.Movies.ToList());
        }

        //
        // GET: /Movies/Details/5

        public ActionResult Details(int id = 0)
        {
            Movie movie = db.Movies.Find(id);
            if (movie == null)
            {
                return HttpNotFound();
            }
            return View(movie);
        }

        //
        // GET: /Movies/Create

        public ActionResult Create()
        {
            return View();
        }

        //
        // POST: /Movies/Create

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(Movie movie)
        {
```

```
    if (ModelState.IsValid)
    {
        db.Movies.Add(movie);
        db.SaveChanges();
        return RedirectToAction("Index");
    }

    return View(movie);
}

//
// GET: /Movies/Edit/5

public ActionResult Edit(int id = 0)
{
    Movie movie = db.Movies.Find(id);
    if (movie == null)
    {
        return HttpNotFound();
    }
    return View(movie);
}

//
// POST: /Movies/Edit/5

[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit(Movie movie)
{
    if (ModelState.IsValid)
    {
        db.Entry(movie).State = EntityState.Modified;
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    return View(movie);
}

//
// GET: /Movies/Delete/5

public ActionResult Delete(int id = 0)
{
    Movie movie = db.Movies.Find(id);
```

```
            if (movie == null)
            {
                return HttpNotFound();
            }
            return View(movie);
        }

        //
        // POST: /Movies/Delete/5

        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Movie movie = db.Movies.Find(id);
            db.Movies.Remove(movie);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

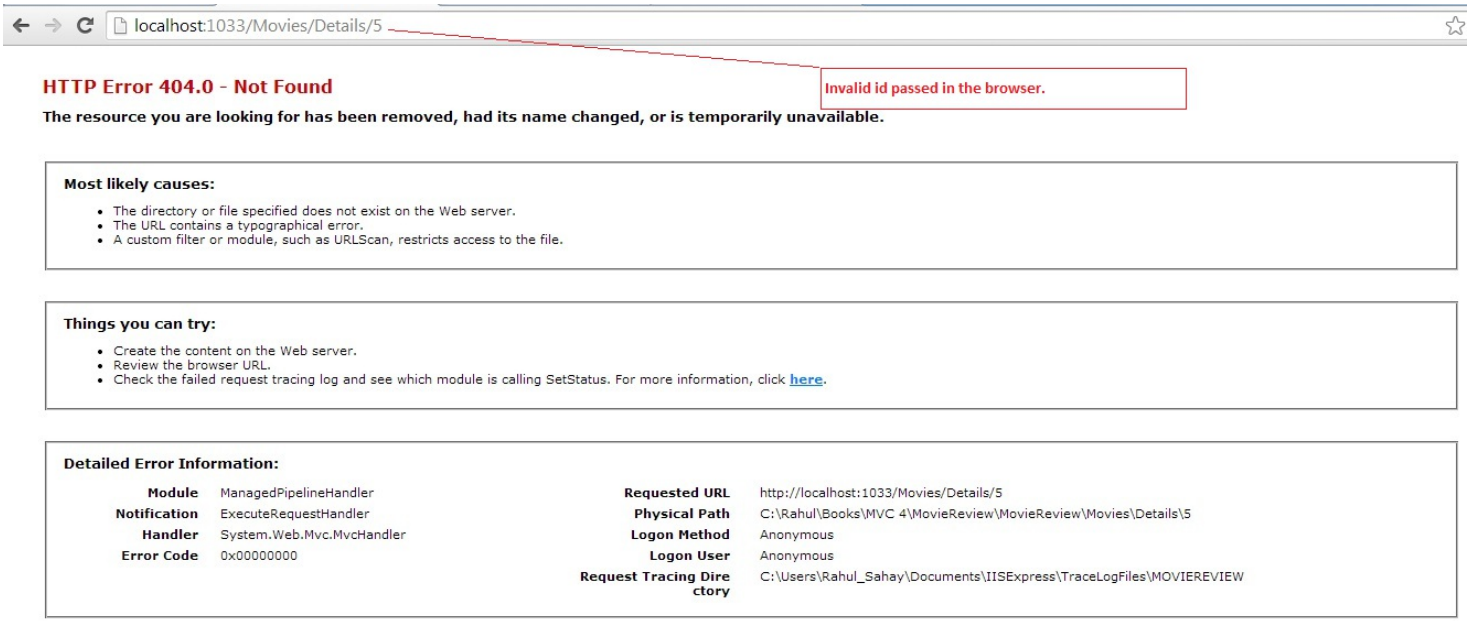
        protected override void Dispose(bool disposing)
        {
            db.Dispose();
            base.Dispose(disposing);
        }
    }
}
```

Now, if you look at the controller code, you could easily see that it instantiated "***MovieDb***" for me, gave me an Index action which has shown all the movies. Associated to the database connection, you could see that it also gave me one "***Dispose***" method to make sure things got cleaned up. So, basically the idea behind this Scaffolding or boiler plate code is that it gives you a very good starting point. From here, you could obviously jump start into an application and test your various scenarios. One point to understand here, that you cannot write complete CRUD operation by just selecting the scaffolding template for any enterprise application. In real world scenario, there are very few applications

which are that simple.

But, it does give a simple codebase to get started. So, right now I am not going delve into each of these controller's action as we will see one by one as we progress in the application. But, just for the sake of understanding we could discuss one Controller action here that is "***Detail***" action. So, it's also very simple method written over here, it will look for the id of the movie and then using the "**Movies**" Dbset, it will fetch the information from the database. So, it will return the record based on the "***MovieId***" and if it doesn't find the Movie in database, then it will simply return 404 error message. Now, let's

check the same by tweaking the value in the browser.



So, if you can see in the above screenshot, I have requested for the "***id(5)***" which doesn't exist and hence, it returned 404 message to the browser. But, obviously you could go ahead and override these kind of browser message by simply creating one view and returning the same if id not found.

Reviews Listing:-

Now, let's go ahead and think of the options of listing the reviews link against the movie associated. So, what I am going to do here is; I am going to introduce one more link in the index page so that it will list Reviews for the associated movie. Now, with the below snippet in place it will simply grab the reviews associated with the Movie.

```
@model IEnumerable<MovieReview.Models.Movie>

@{
    ViewBag.Title = "Index";
}

<h2>Index</h2>

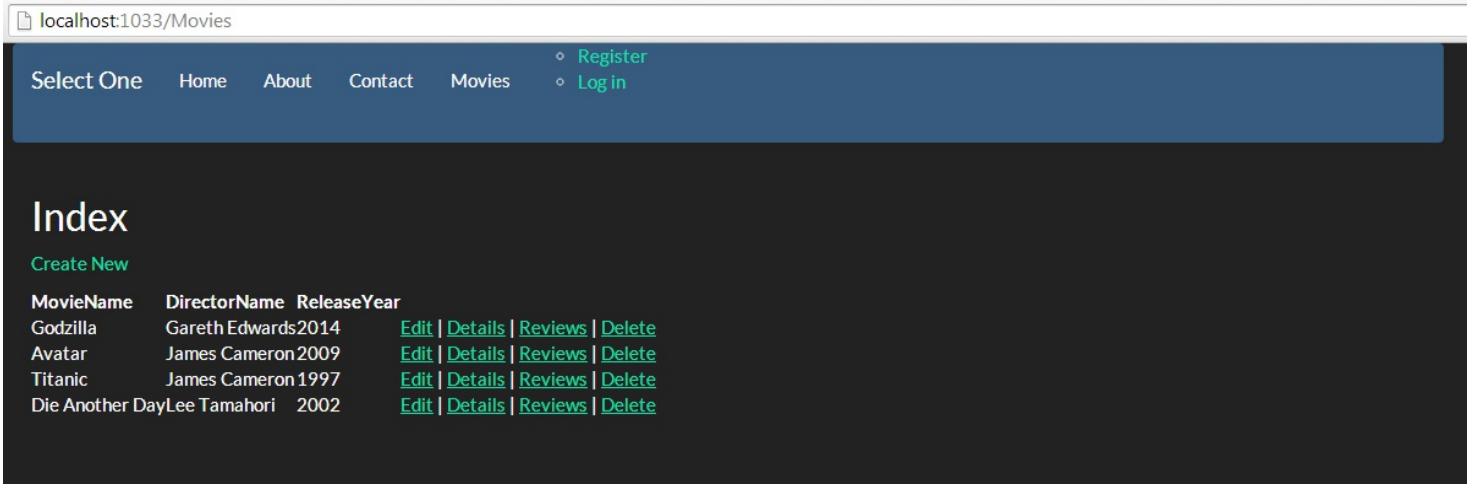
<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table>
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.MovieName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.DirectorName)
```

```
        </th>
        <th>
            @Html.DisplayNameFor(model => model.ReleaseYear)
        </th>
        <th></th>
    </tr>

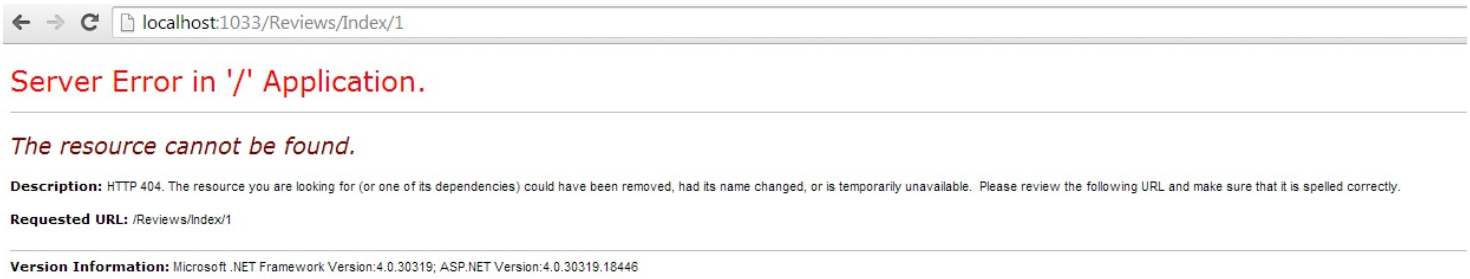
@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.MovieName)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.DirectorName)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.ReleaseYear)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.Id }) |
            @Html.ActionLink("Details", "Details", new { id=item.Id }) |
            @Html.ActionLink("Reviews",  "Index",  "  MovieReviews",  new  {  id  =
item.Id }, null) |
            @Html.ActionLink("Delete", "Delete", new { id=item.Id })
        </td>
    </tr>
}

</table>
```

So, now, when I refresh the page then I can see the newly generated link appear on the page.



But, when I click on the associated Reviews Link, it will produce me the below 404 error for obvious reasons because I haven't modified the commented code in the controller yet.



So, now let's go ahead and uncomment the "***MovieReviews***" controller code and

delete the unexpected action methods. So, at this point of time, I need only index action. But, we do need our context here. So, my final modified controller looks like below snippet.

```
using MovieReview.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace MovieReview.Controllers
{
    public class MovieReviewsController : Controller
    {

        MovieDb _db = new MovieDb();
        //
        // GET: /MovieReviews/

        public ActionResult Index([Bind(Prefix="id"] int movieId)
        {
            var movie = _db.Movies.Find(movieId);
            if(movie!=null)
            {
                return View(movie);
            }
            return HttpNotFound();
        }

        protected override void Dispose(bool disposing)
        {
            _db.Dispose();
            base.Dispose(disposing);
        }
    }

}
```

So, as you could see above in the snippet I have declared one variable for my context to instantiate the "***MovieDb***". And since I already have datacontext declared here; hence, I have also implemented dispose method to clean the things up. Here, I have used Bind mechanism to bind parameter. Now, what bind is doing here; it tells the MVC Model binder that when it goes to look for "***MovieId***" parameter value, look for something called "***id***". Now, the next lines are pretty obvious, it will look for movie based on the "***MovieId***" passed in. If it's not equal to null, we will simply return view associated for that otherwise, will simply return 404 error. But, at this point I already have a view,

which just needs little bit of modification.

```
@model MovieReview.Models.Movie

@{
    ViewBag.Title = "Index";
}

<h2>Review for @Model.MovieName</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>

@Html.Partial("_Reviews",Model.reviews)
```

So, if you notice the above snippet, I have split the Movie Review index into partial view which is going to render me a table based on the reviews getting

passed in from here. Now, I have created one partial view with the name "***_Reviews***" with the following piece of code.

```
@model IEnumerable<MovieReview.Models.MovieReviews>
<table>
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.ReviewerName) |
        </th>
        <th>
            @Html.DisplayNameFor(model => model.ReviewerComments) |
        </th>
        <th>
            @Html.DisplayNameFor(model => model.ReviewerRating)
        </th>
        <th></th>
    </tr>

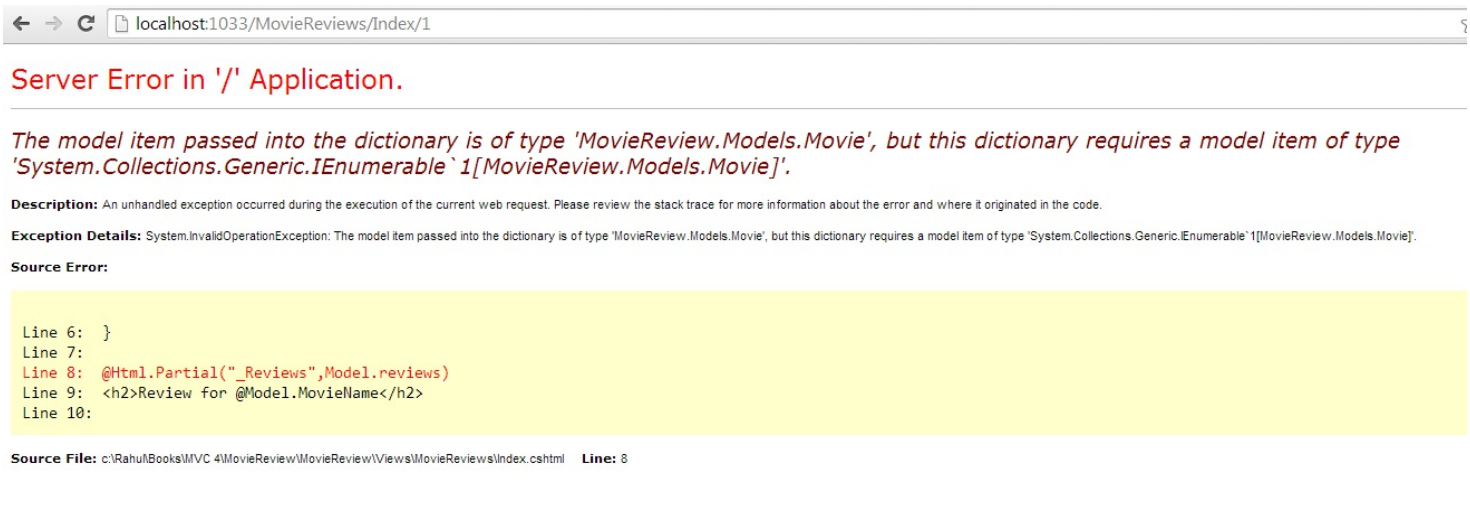
    @foreach (var item in Model)
    {
```

```
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.ReviewerName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.ReviewerComments)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.ReviewerRating)
            </td>
            <td>
                @Html.ActionLink("Edit", "Edit", new { id = item.Id })

            </td>
        </tr>
    }

</table>
```

Now, let's go ahead save everything, build the application and then try this thing out.



Now, the problem over here is associated with the Entity Framework because here reviews property by default will be null. So, when you try to do "***Movies.Find(id)***" in the controller, it will simply return the Movie Entities properties, it won't touch "***Reviews***" because this particular information is in different table. Hence, here there is **One to Many Relationship** is associated between "***Movie***" and its "***Reviews***" and **EF** couldn't load these child collections automatically.

So, in order to fix this problem, I'll just go ahead and add "***Virtual***" keyword in movie class before the collection, which will tell EF to extract the "***Reviews***"

associated for that "***Movie***" also. So behind the scenes there will be two "***db queries***", one to load the movie and another one to load the Reviews. However, this solution won't be the best way to solve the problems in many big enterprise projects. But, in that case we could make use of LINQ concept "**EagerlyLoading**" which will not only load the Movies information but also reviews associated with this in one shot.

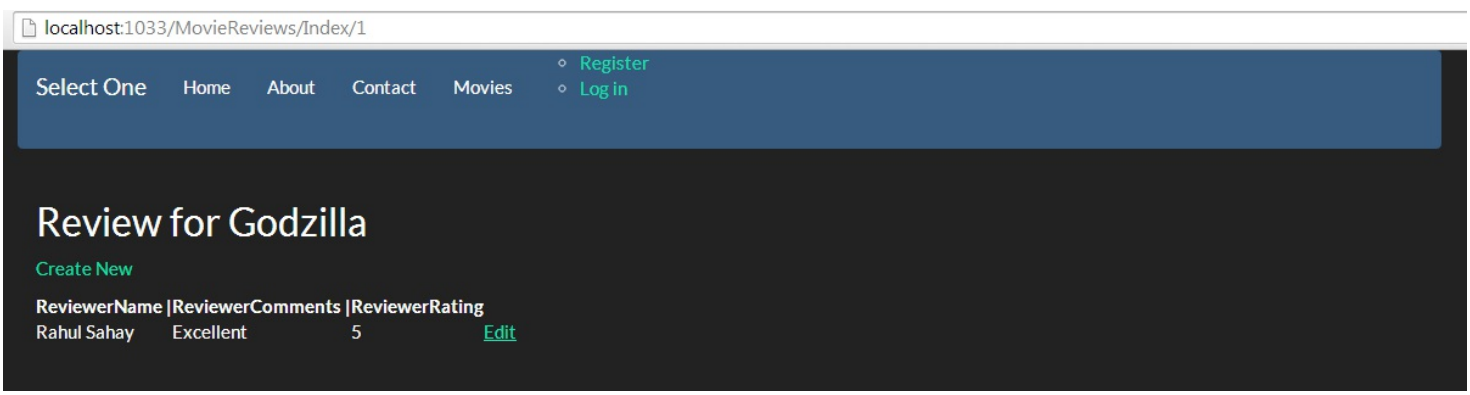
You can refer the same on **MSDN LINQ** article. But, for now let's assume that by placing virtual in front of reviews will solve the problem for me. So, my modified Movie class looks like

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace MovieReview.Models
{
    public class Movie
    {
        public int Id { get; set; }
        public string MovieName { get; set; }
        public string DirectorName { get; set; }
        public string ReleaseYear { get; set; }

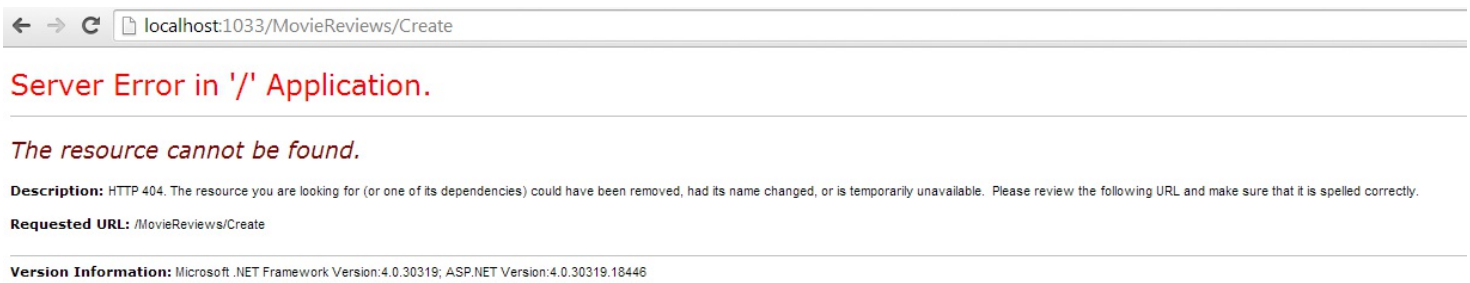
        public virtual ICollection<MovieReviews> reviews { get; set; }
    }
}
```

Now, with the above piece of code in place, let's go ahead build the solution and refresh the page, it will now give me the desired result.



Create and Edit Facility of Review:-

Now, let's turn our attention towards creating review. Now, when I click on the link Create New, it will give me the below error which is obvious.



Now, let's fix this. So, in order to fix the same I need to modify my index page code. As you can see in the below modified view I have added a parameter that will tell the routing engine to please

pass a parameter called "**movieId**" assigned with "***Model.Id***".

```
@model MovieReview.Models.Movie


@{
    ViewBag.Title = "Index";
}

<h2>Review for @Model.MovieName</h2>

<p>
    @Html.ActionLink("Create New", "Create", new { movieId=Model.Id})
</p>

@Html.Partial("_Reviews",Model.reviews)

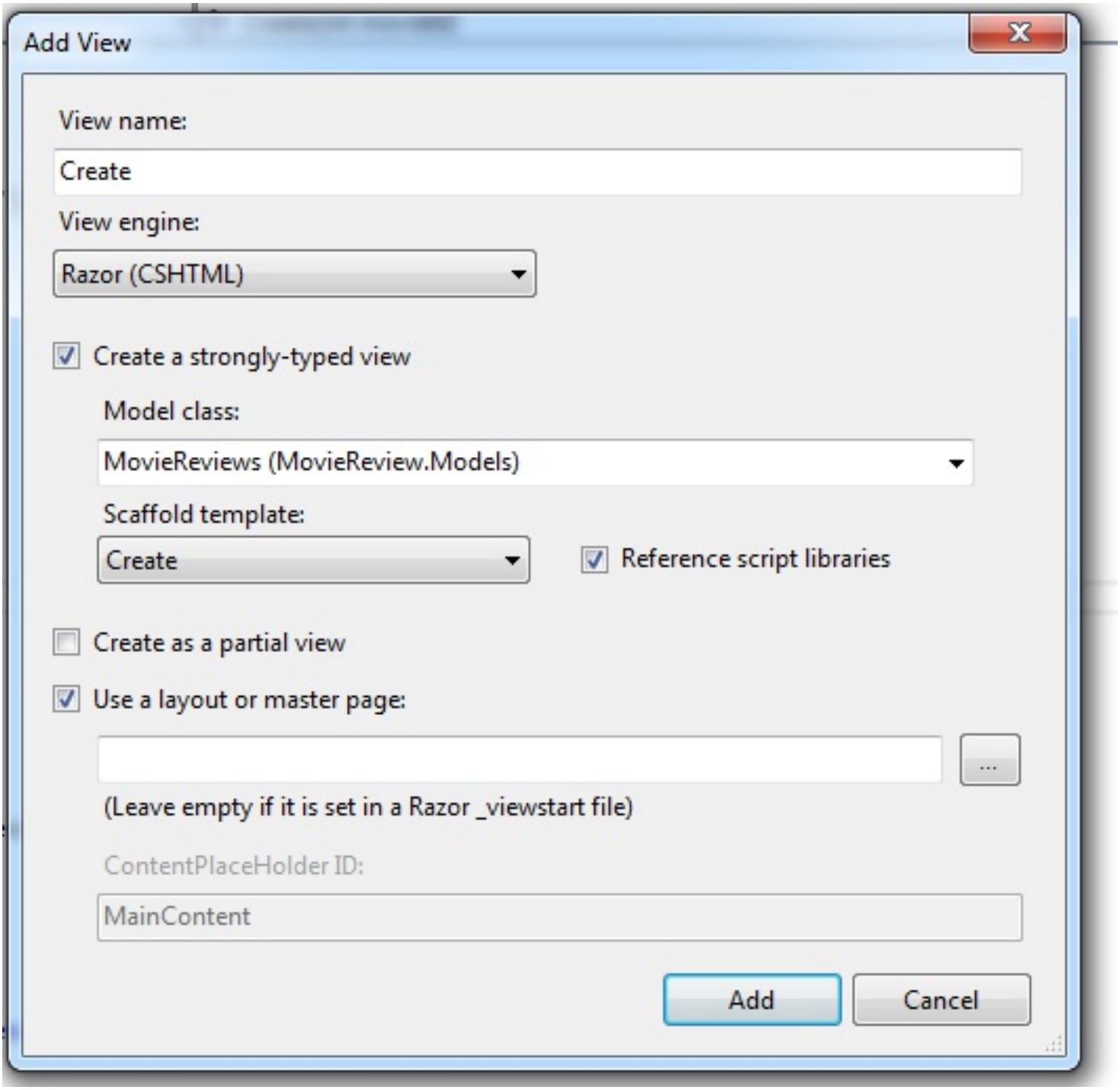
<div>
    @Html.ActionLink("Back to List", "Index","Movies")
</div>
```

Now, what the above piece of code will do? It will give us a proper link. Now, in the "***MovieReviews***" controller, I need the action which will respond to the "***get***" request of the "***Create***".

Below, as you can see; this is a very simple action method which simply returns a view based on the incoming "***Id***". However, for creating the view we obviously won't need this parameter, but I keep it for future use.

```
[HttpGet]
        public ActionResult Create(int movieId)
        {
            return View();
        }
```

However, I do need to create one view for the same. So, I just right-click on the index method and Add a view with the below mentioned parameters.



Now, when I go ahead and click on Add it will generate me the desired view. But, let me fix the UI of this View. So, ideally this view will look like

```
@model MovieReview.Models.MovieReviews

@{
    ViewBag.Title = "Create";
}

<h2>Create</h2>

@using (Html.BeginForm()) {
    @Html.AntiForgeryToken()
    @Html.ValidationSummary(true)
<form class="form-horizontal">
    <fieldset>
        <legend>MovieReviews</legend>
```

```
        <div class="form-group">
            @Html.LabelFor(model => model.ReviewerName)

            <div class="form-control">
                @Html.EditorFor(model => model.ReviewerName)
                @Html.ValidationMessageFor(model => model.ReviewerName)
            </div>
        </div>
        <div class="form-group">
            @Html.LabelFor(model => model.ReviewerComments)

            <div class="form-control">
                @Html.EditorFor(model => model.ReviewerComments)
                @Html.ValidationMessageFor(model => model.ReviewerComments)
            </div>
        </div>
        <div class="form-group">
            @Html.LabelFor(model => model.ReviewerRating)

            <div class="form-control">
                @Html.EditorFor(model => model.ReviewerRating)
                @Html.ValidationMessageFor(model => model.ReviewerRating)
            </div>
        </div>

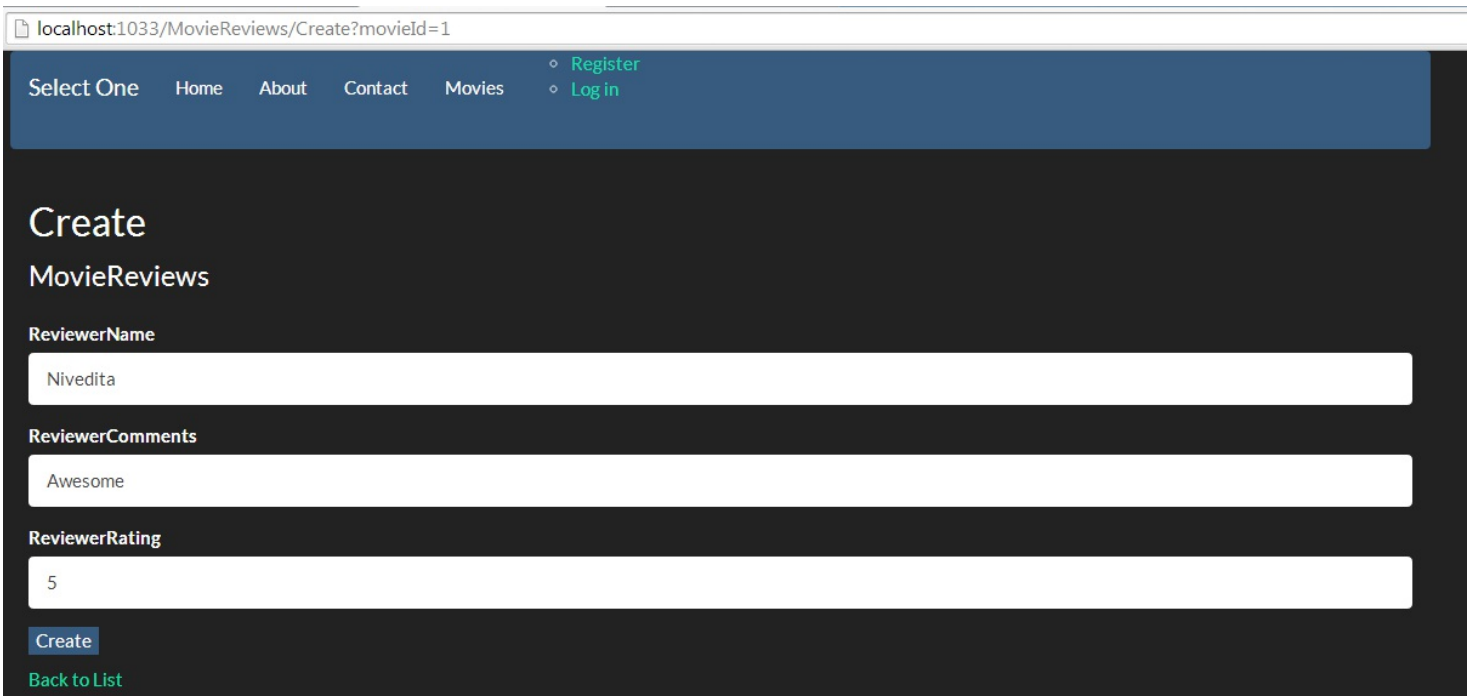


        <p>
            <input class="btn-primary" type="submit" value="Create" />
        </p>
    </fieldset>
    </form>
}

<div>
    @Html.ActionLink("Back to List", "Index","Movies")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

So, as you can see in the above piece of code that it is using HTML helper "***BeginForm***" which will **postback** to the same form when user click on the save button. Now, when I build and refresh the page, I should be getting my desired page at the same place.



Now, in here I need one more action to respond to our Create action call. And

this action method will look like this.

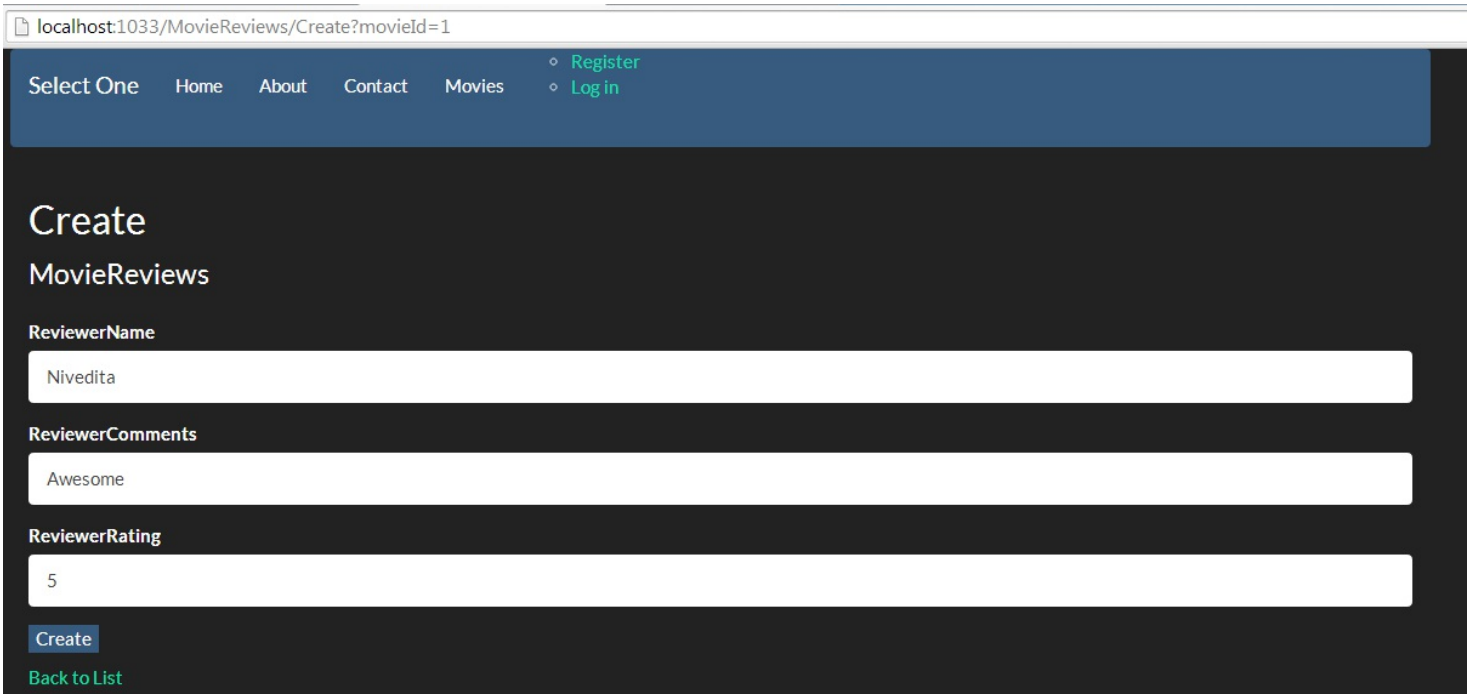
```
[HttpPost]
        public ActionResult Create(MovieReviews _review)
        {
            if(ModelState.IsValid)
            {
                _db.MovieReviews.Add(_review);
                _db.SaveChanges();
                return RedirectToAction("Index", new { id = _review.MovieId });
            }
            return View(_review);
        }
```

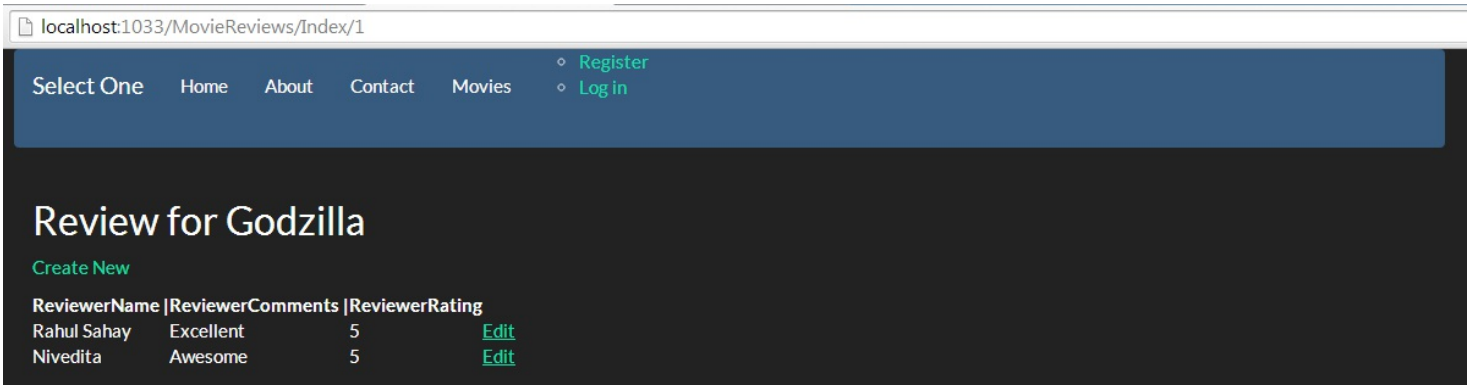
So, what will happen here, this method will respond to the "***Post***" request which will take "***MovieReviews***" as a parameter and then **Model Binder** from the MVC will come into play and instantiate an instance of this type and then look all the places to find the properties associated with that and then it will push these values into it. Now inside the Create action it will first

check whether the "***ModelState***" flag is valid or not. If this flag returns false, then it means something went wrong while saving the same. We'll go through these validation rules in the next section, but for now just understand that if due to any reason validation fails then it will simply return the same view forcing the user to fix the same and then save the changes.

On the other hand, if everything is valid, then in that case, "***ModelState***" flag will be true and then it will add the new review to the database, save the same and then redirect to the reviews page, so that I can see them., Now everything is in place, let's go ahead build the

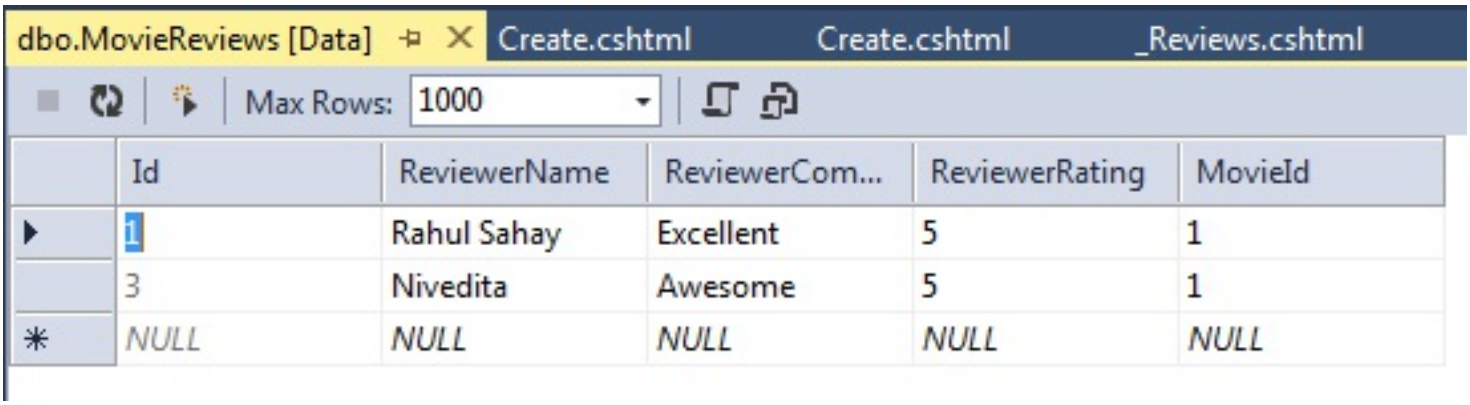
application and then try again.





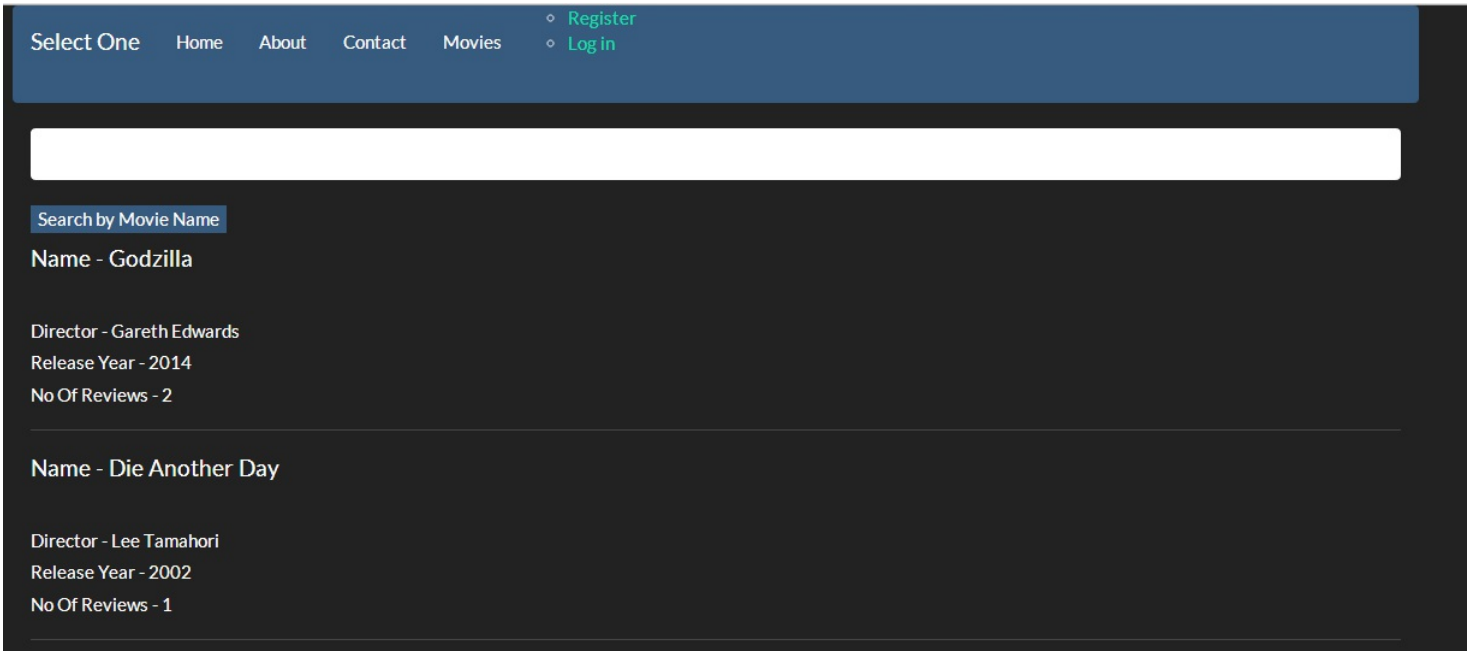
So, as we can see in the above screen shot after saving the same it get redirected to the Reviews page for the particular movie. Now, let's go ahead

and verify the same from the database.



Id	ReviewerName	ReviewerCom...	ReviewerRating	MovieId
1	Rahul Sahay	Excellent	5	1
3	Nivedita	Awesome	5	1
NULL	NULL	NULL	NULL	NULL

Let's go ahead and implement the edit part as well. But, before that let's see the Home page once, now here number of reviews should have been updated.

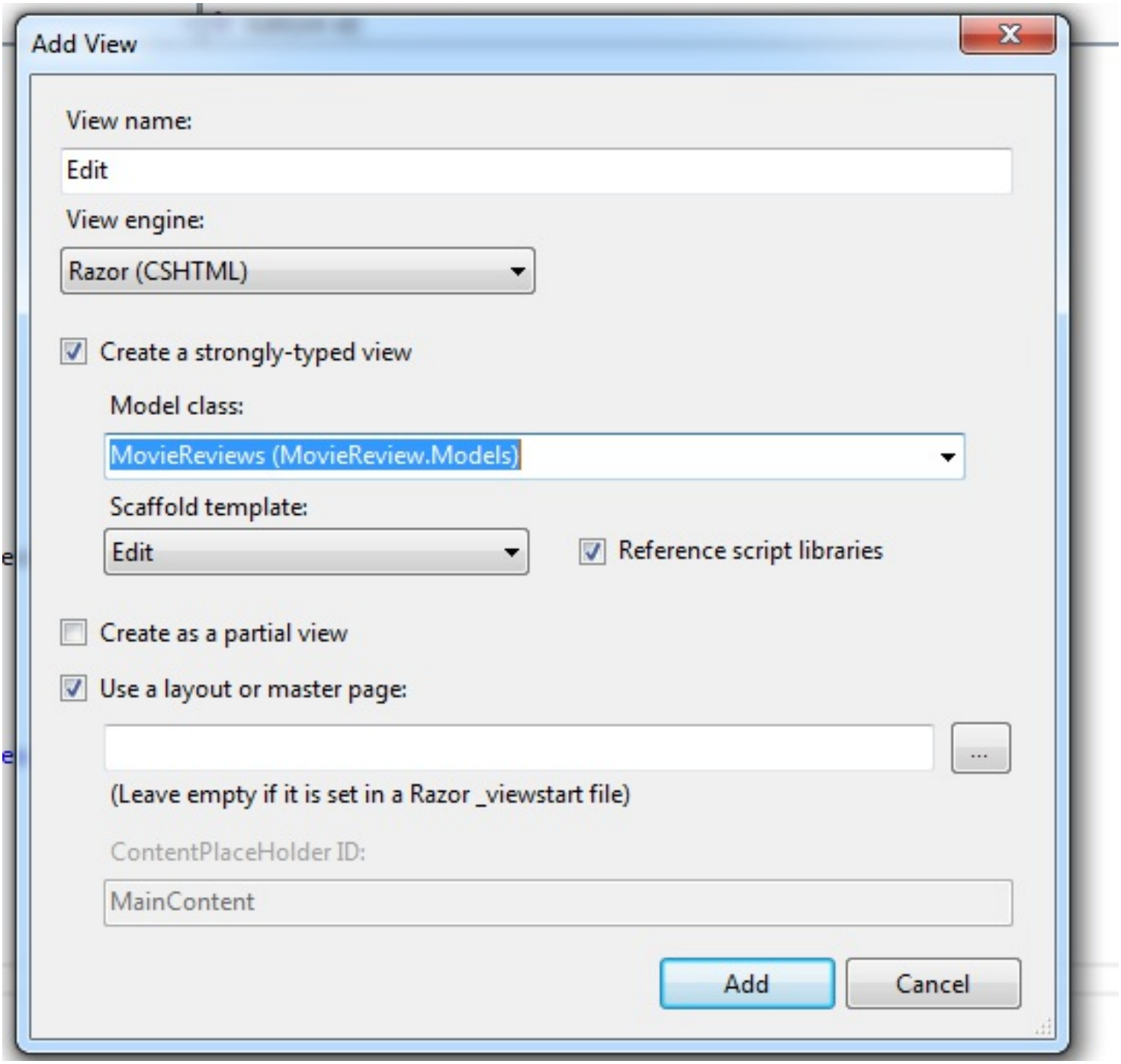


Now, let's talk about editing the review. So, here also for implementing the Edit action the course of action will remain same, 1st we need an action method which will respond to the "***get***" request of "***Edit***" method. So, my edit action method will look like

```
[HttpGet]
        public ActionResult Edit(int id)
        {
            var model = _db.MovieReviews.Find(id);
            return View(model);
        }
```

So, let me explain for a second that what the above piece of code is doing. Here, it is taking **id** as input parameter and looking for review based on the incoming parameter. So, it will return

the user that particular review which will allow the user to edit that. Now, to respond to this "***get***" request, I need a view. But it seems like I already have a view in there, just modify the same. But, it would be good idea to create a fresh template, so delete the existing View and add one.



So, in the above screen shot as you can see that, I have selected "***MovieReviews***" as an Edit scaffolding template, so it will emit me the template

with Edit functionality. So, the modified view would look like something shown below.

```
@model MovieReview.Models.MovieReviews

@{
    ViewBag.Title = "Edit";
}

<h2>Edit</h2>

@using (Html.BeginForm()) {
    @Html.AntiForgeryToken()
    @Html.ValidationSummary(true)
<form class="form-horizontal">
```

```
    <fieldset>
        <legend>MovieReviews</legend>

        <div class="form-group">
            @Html.HiddenFor(model => model.Id)
            @Html.LabelFor(model => model.ReviewerName)
            <div class="form-control">
                    @Html.EditorFor(model => model.ReviewerName)
                    @Html.ValidationMessageFor(model => model.ReviewerName)
                </div>
            </div>
        <div class="form-group">
            @Html.LabelFor(model => model.ReviewerComments)

            <div class="form-control">
                @Html.EditorFor(model => model.ReviewerComments)
                @Html.ValidationMessageFor(model => model.ReviewerComments)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.ReviewerRating)

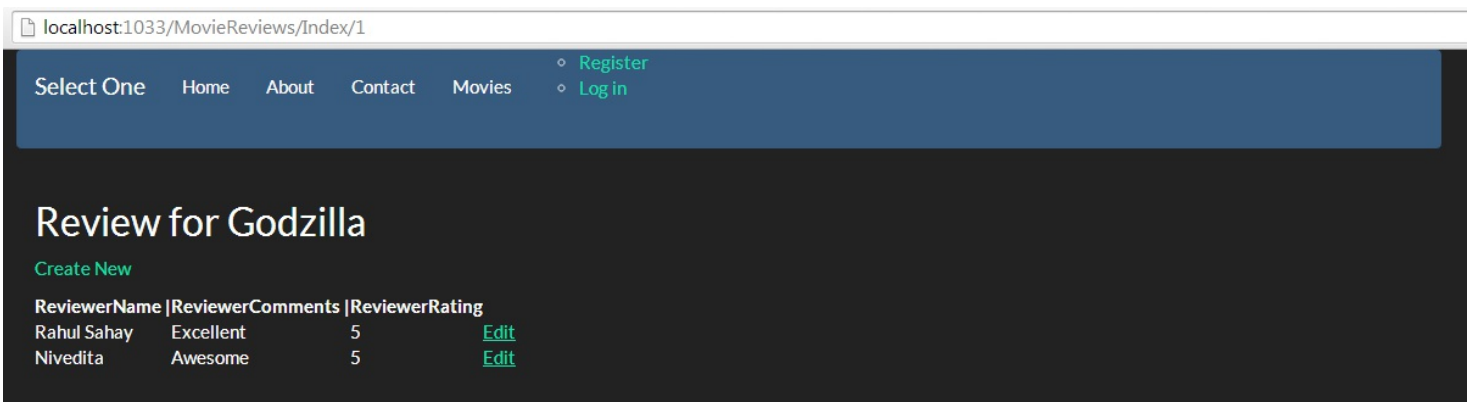
            <div class="form-control">
                @Html.EditorFor(model => model.ReviewerRating)
                @Html.ValidationMessageFor(model => model.ReviewerRating)
            </div>
        </div>

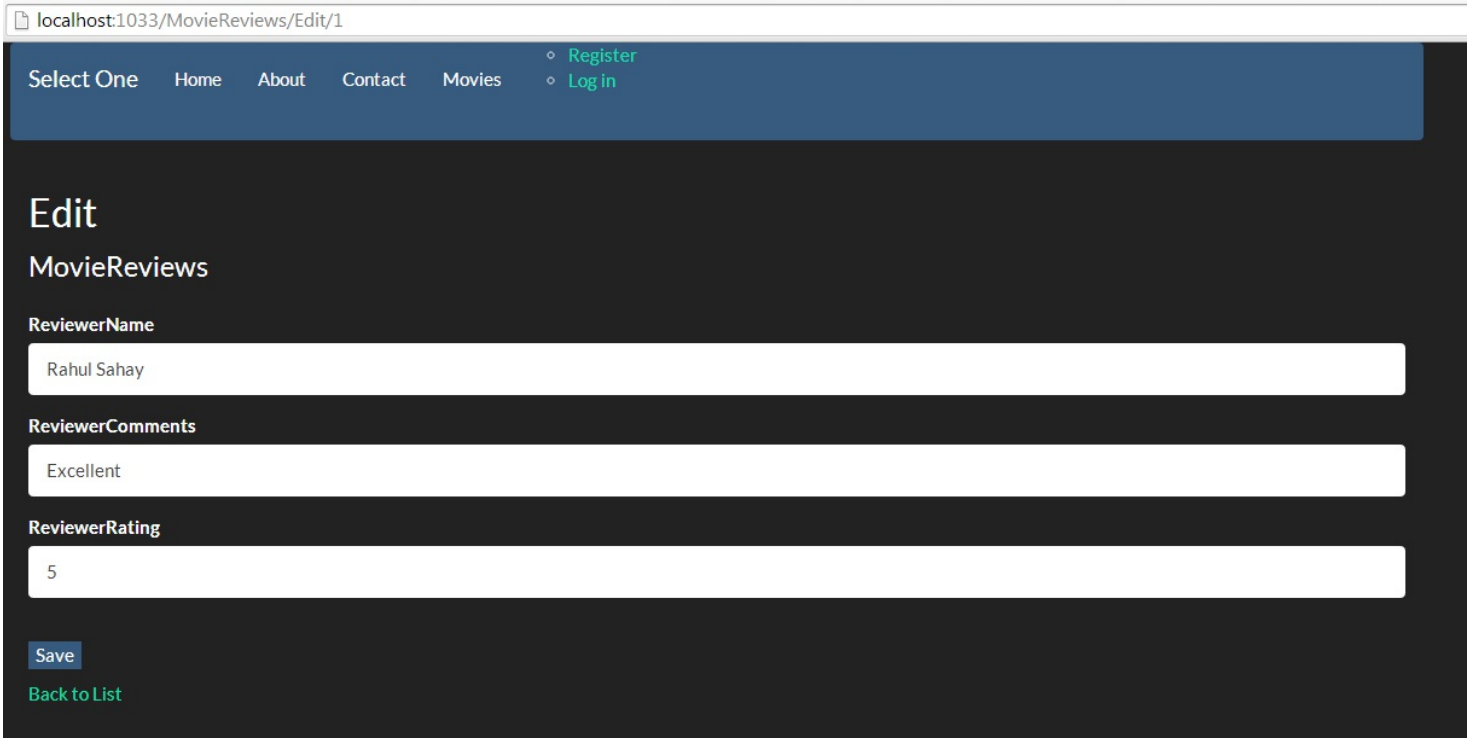
        @Html.HiddenFor(model=>model.MovieId)
          <p>
              <br />
                <input class="btn-primary" type="submit" value="Save" />
            </p>
</fieldset>
    </form>
}

<div>
@Html.ActionLink("Back to List", "Index", new { id=Model.MovieId})
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

So, when you see the above piece of code you will see that I have used one HTML helper "***HiddenFor***", this is just to ensure that we are carrying the id value associated with that "***movie***" and "***movieReview***" for editing the review. But, since it's a key, it doesn't require user to change that, hence it's masked. So, now when we click on the edit link for any review as shown below, it will emit me the required view in the edit pattern.





So, as you can see in the above screen shot based on the **id** which we passed from the action method, it fetched me the view in edit pattern. Now, when I hit save, I do need an action method which will respond to this request. So, this action method is obviously going to be the "***Post***" request. And this will look like as shown below.

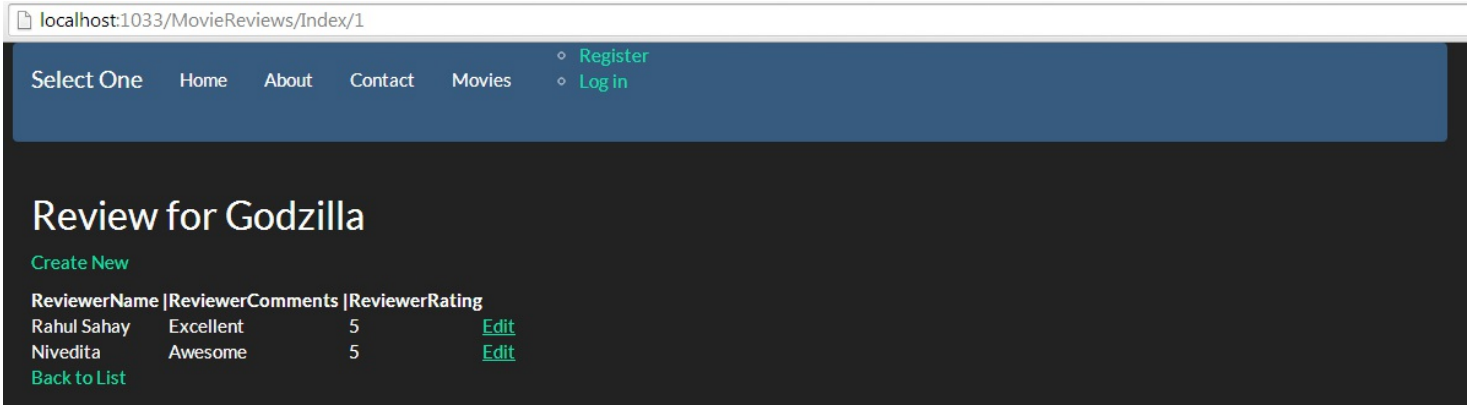
```
[HttpPost]
public ActionResult Edit([Bind(Exclude="ReviewerName")]MovieReviews review)
{
    if (ModelState.IsValid)
    {
        _db.Entry(review).State = EntityState.Modified;


        _db.SaveChanges();
        return RedirectToAction("Index", new { id = review.MovieId });
    }
    return View();
}
```

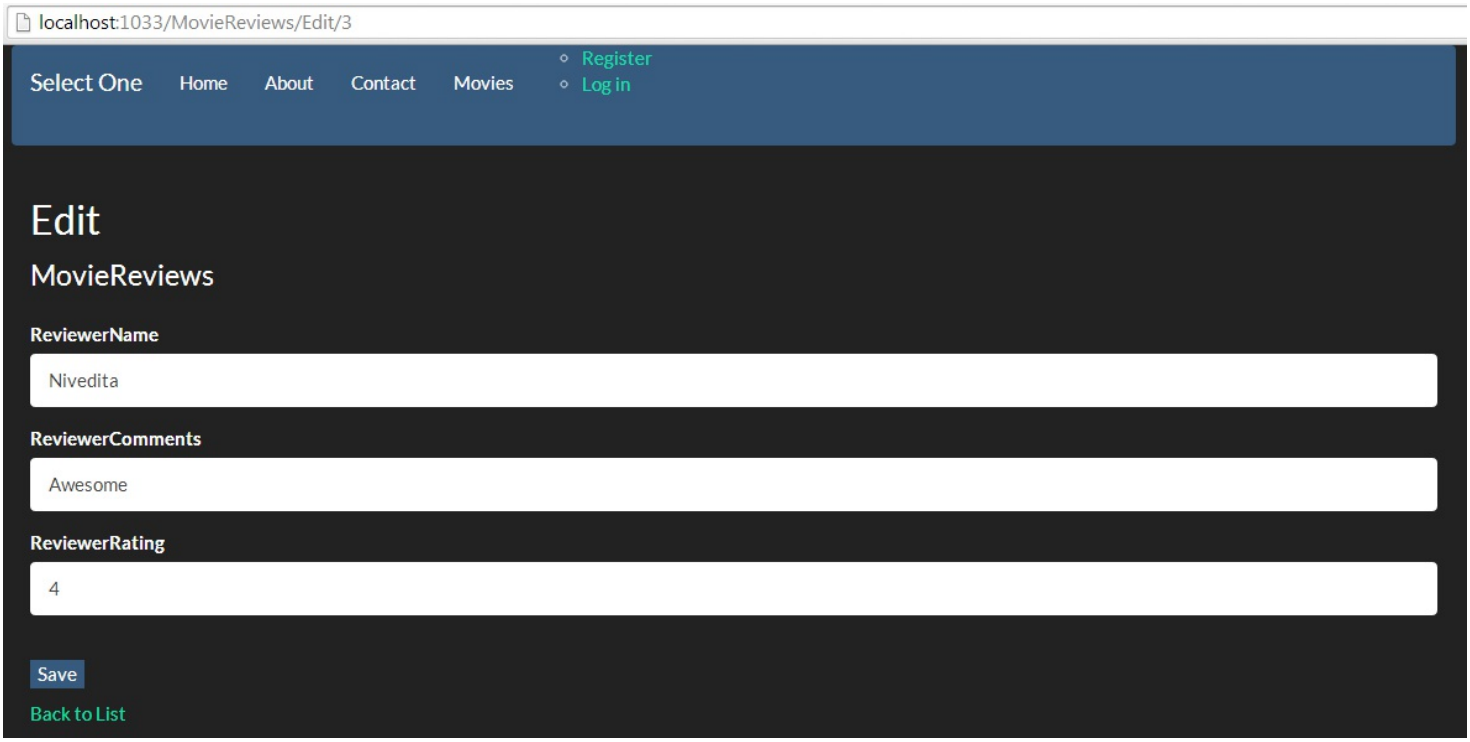
So, here it is again going to take "***MovieReviews***", again the MVC Model binder is going to populate the object with the things it finds in the request. So, here we don't have to look for posted form values or query strings, the model binder will take care for this. Here, again the check for "ModelState.IsValid", simply return the same view in case model state is not valid. But, if it is valid, then our

approach here will be little different.

Property "***EntityState***" will be available in the "***System.Data***" namespace. Here, the "***Entry***" is way to tell the EF, that here is a review that I want to modify. So, if there is a value change in it, then EF will go ahead and update the record in the database. Here what we did is we attach an existing review with that entry; set the state to "***Modified***" and then **EF** will update the record. Again, on successful, change we'll redirect the user back to index action. So, let's go ahead and build the app and see whether it's taking effect or not.

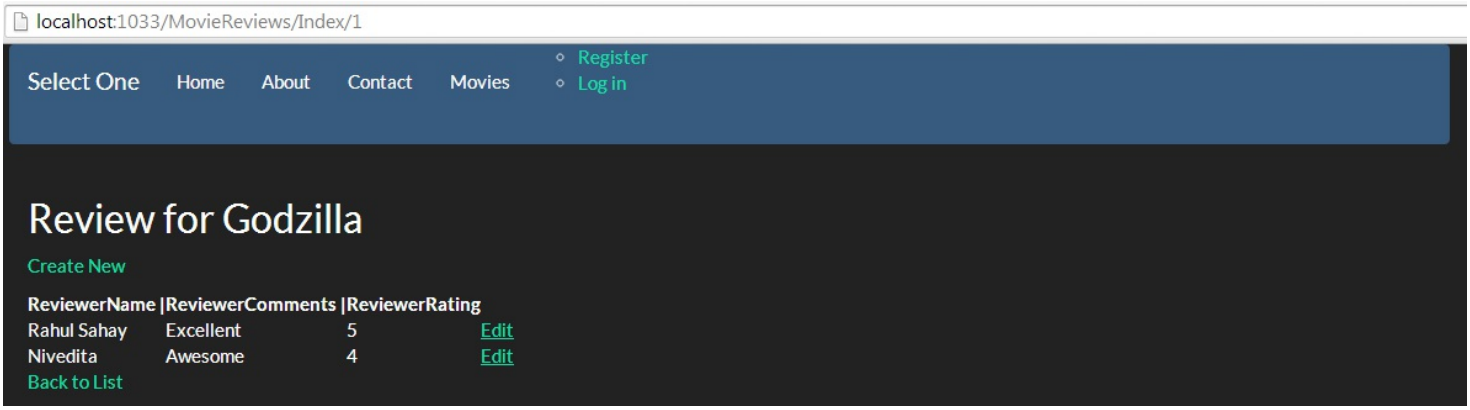


Down, in the below screenshot, I have changed the rating to 4.



Now, let me go ahead and click on save and check whether the same is working

fine or not.



And it did work. Now, if you have noticed the above code review that I did include "([Bind(Exclude="ReviewerName")]" in my method. The only reason behind this I don't want user to edit the reviewer name. There are number of ways to fix this issue, but this approach is the quickest and easiest way to fix the same. However, you might be thinking if you delete the "***ReviewerName***" name

portion from the view, user cannot edit the same as he/she can't see it,

But obviously user may simply issue a get request from the browser and update the record. But, this "***Exclude***" will act as a **blacklist**, so any value is getting passed on to this property will be not considered. So, you need to use the same in conjunction with **View** changes. Then from View you need to delete the "***ReviewerName***" so that no one can even attempt that. But, for now I don't need that hence, I am removing this Bind part from the method and build again.

Validation Annotations:-

Now, the next thing which I want to discuss here is different kind of validations associated with MVC. Now, I would like to restrict a user to enter only valid rating. Currently, user can add any rating which user wants. But, with the implementation of Range, user will be restricted.

```
public class MovieReviews
    {
        public int Id { get; set; }
        public string ReviewerName { get; set; }
        [Required]
        [StringLength(200)]
        public string ReviewerComments { get; set; }
        [Range(1,5)]
        public int ReviewerRating { get; set; }

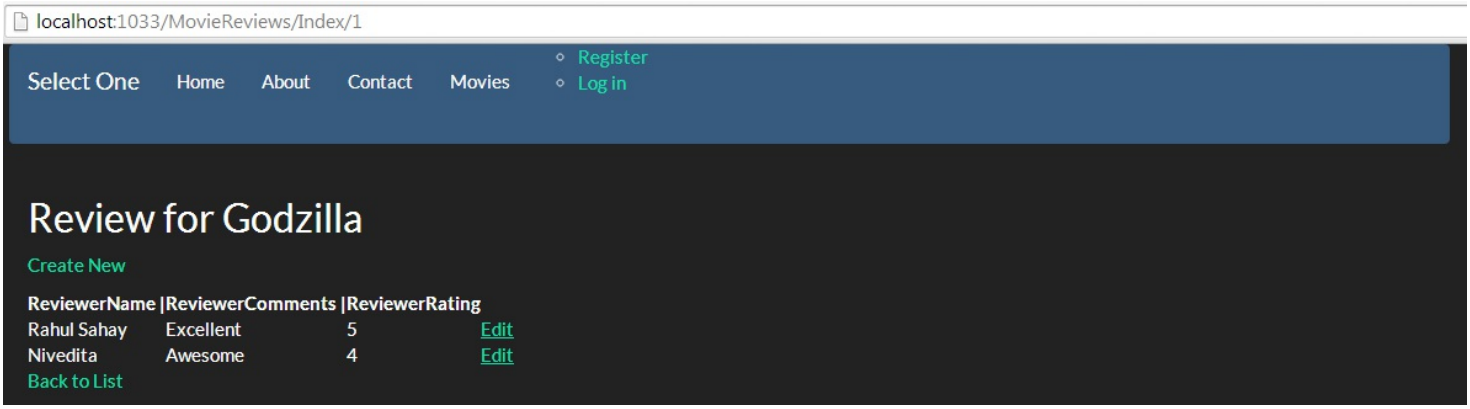
        public int MovieId { get; set; }
    }
```

So, here as you can see I have added the annotation in the "***MovieReviews***" class. And this annotation will be available under the namespace ***"System.ComponentModel.DataAnnota***
So, we need to include this first. Now, when we do model binding to these attributes, Model Binder will also check any validation logic applied to these attributes, if yes then it will implement the same against the value which you enter there.

There's also a "***Required***" property which will tell user that it is mandatory field. You might have wondered that why I have not used "***Required***" for rating, because rating is an integer property and

which is value type, and value type can't be null, hence annotating rating with "***Required***" is a kind of redundant thing here.

I have also restricted max value for the comment by using "***StringLength***" property simply. Apart from these basic validations, there are many more attributes which you can refer like for validating **Regular Expression, Remote Validation** and many more. You can read these attributes in detail on "***MSDN***" Site itself. So, for now I am just focusing on the basic validations. Now, when we build the same try to enter something out of range, then it will behave as shown below.

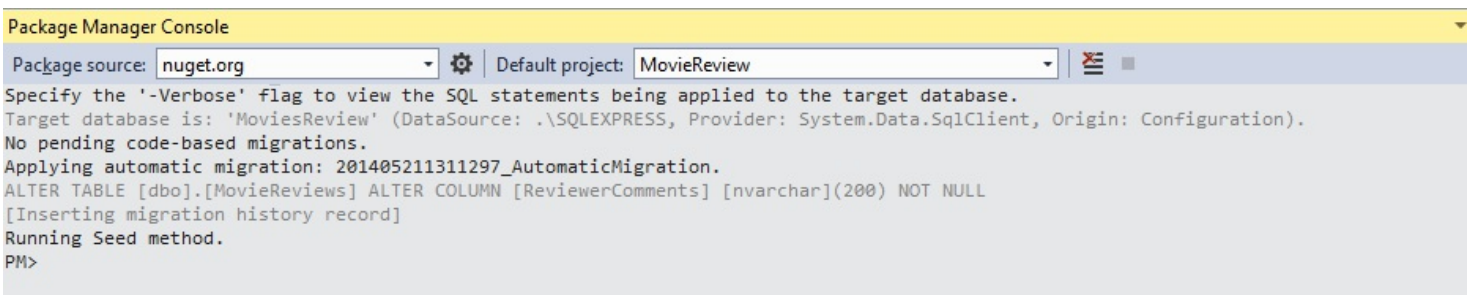


So, now when I click on the "***Edit***" link for Nivedita, it banged and produced the below error.

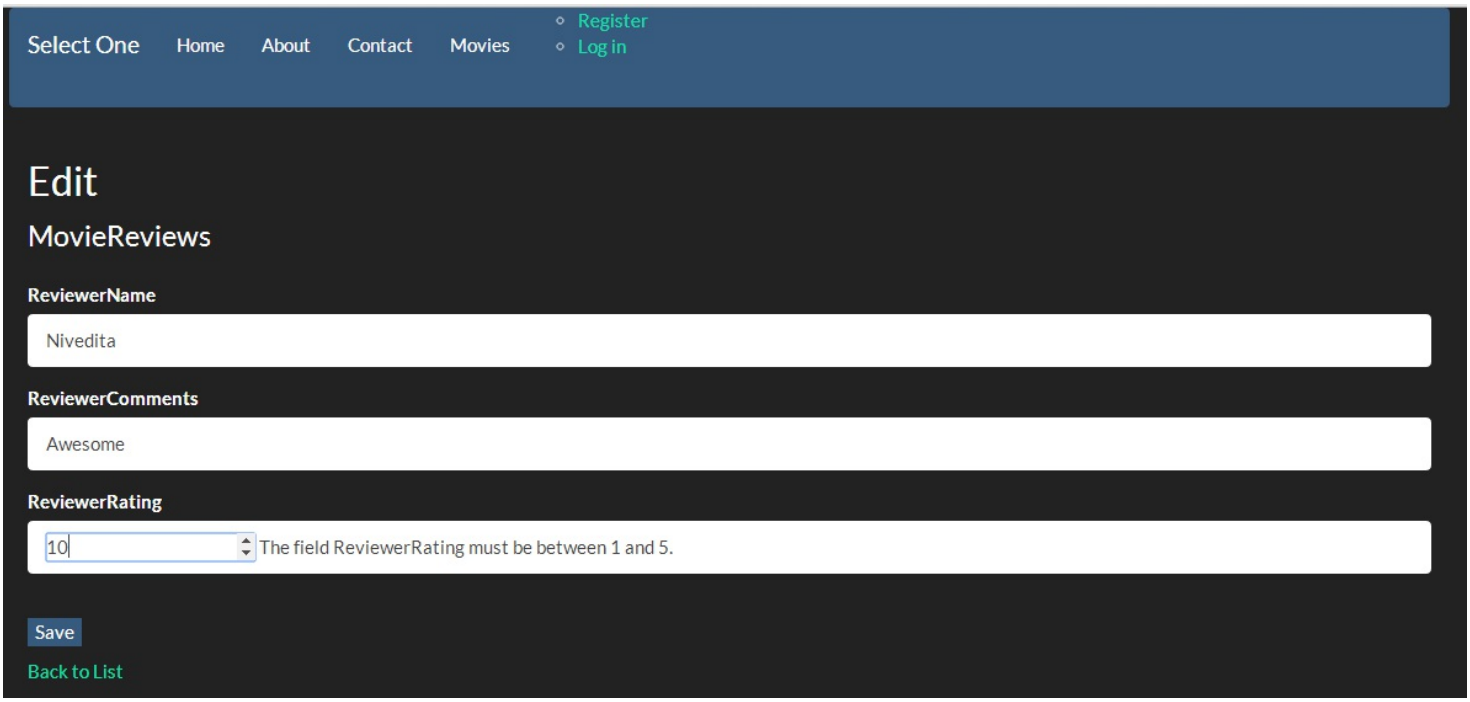


So, the reason for this failure is **EF**. EF is always checking the model against the model currently in use whether they are in sync which you have created at the

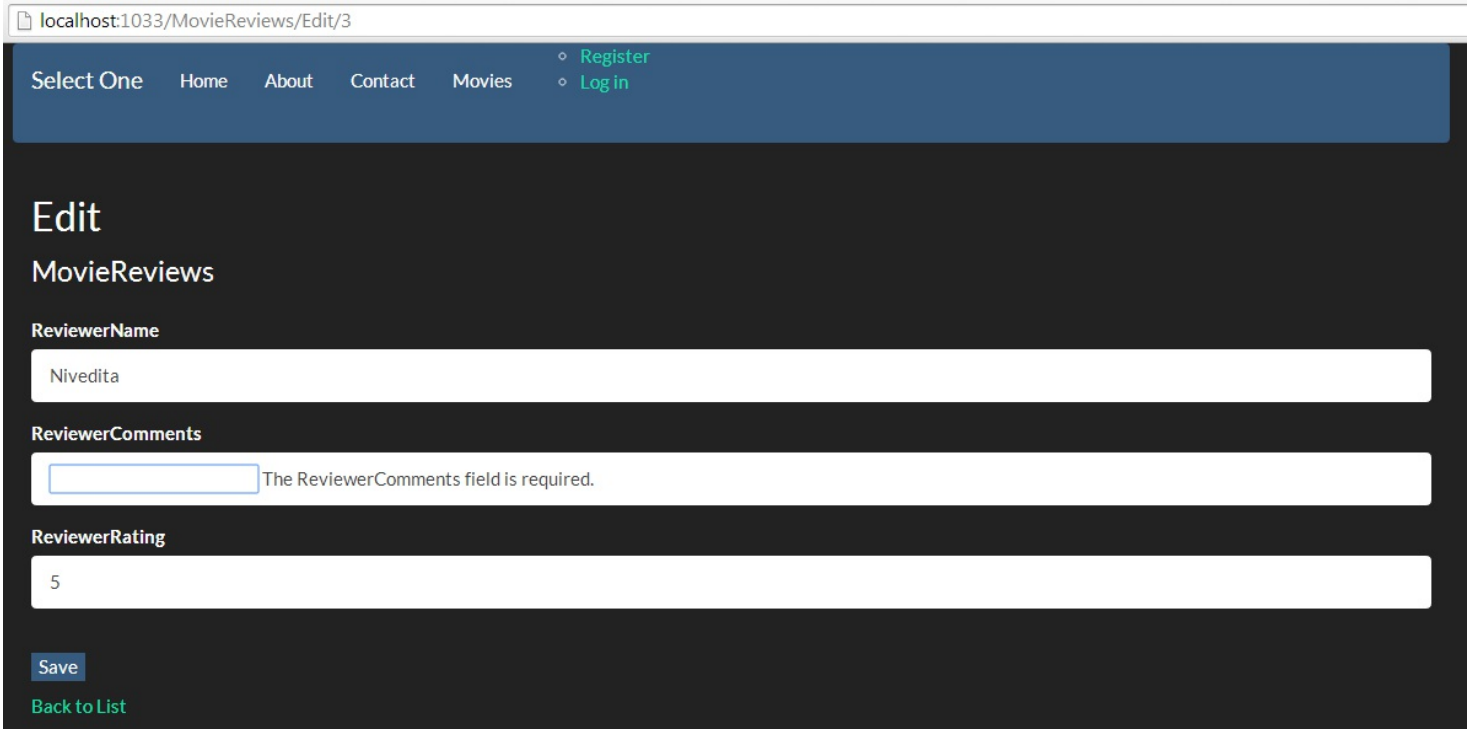
time of migrations. So, here EF detected that model has changed like "***Required***" field which is going to tell the schema that I do need "***NOT NULL***" column. So, the model needs to be updated. Similarly, I have also restricted the size for string length; previously it was "***NVARCHAR MAX***". So, in order to fix the same what I could do here, is just update the database from the powershell command "Update-Database -Verbose -Force" and then it will update the database with the new model.



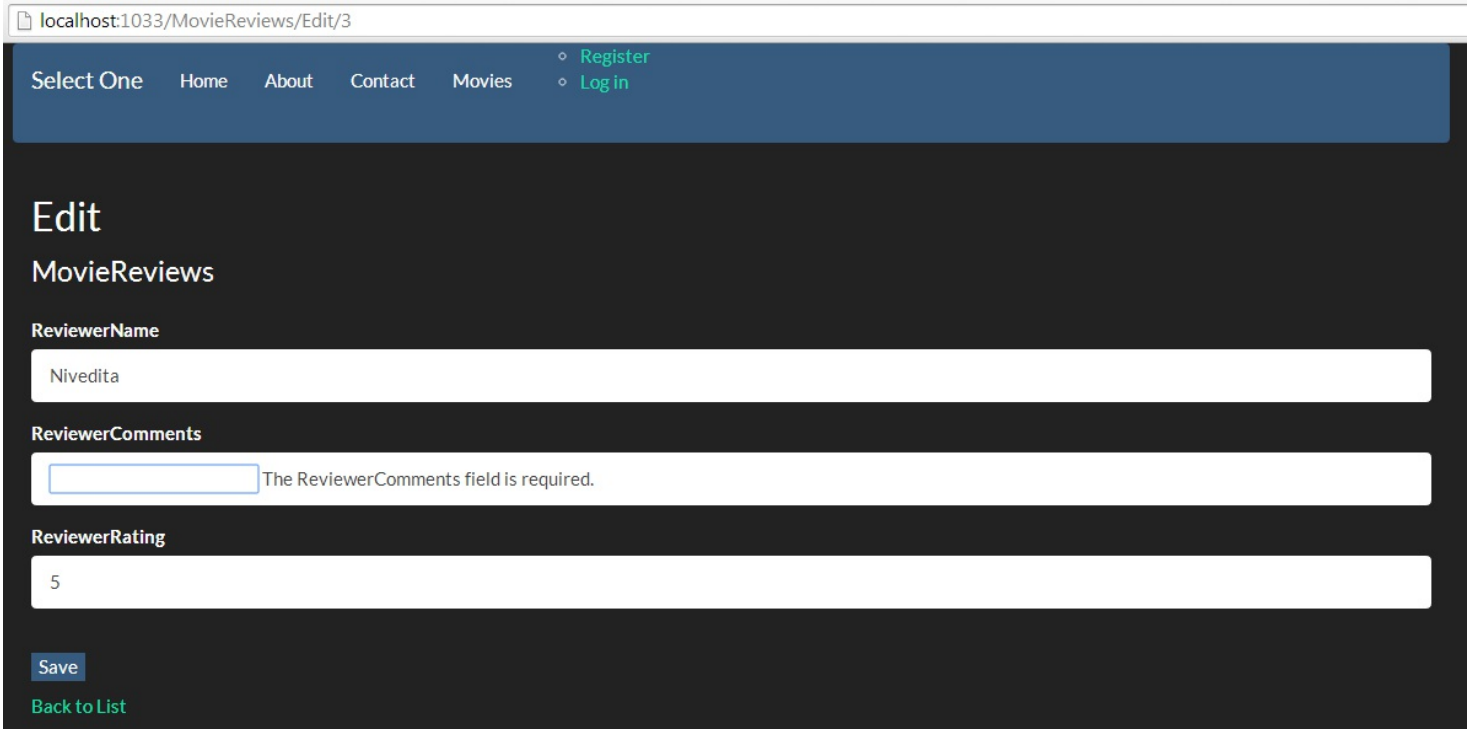
So, up here is the confirmation that model is updated correctly. Now when I refresh the page, it will take me to the required page as shown below.



Now, when I go ahead and try something out of range, then it will give me the below error.



So, as soon as I typed 10, it picked the wrong value and gave me an indication that value you entered is out of range. So, here "***EditorFor***" HTML helper comes into action, it emits the data into the input tag which client side JavaScript picked up and exercised the client side validation for you.



In the above screen shot same thing happened with the Comments field. So, even if user tries to click the save button, nothing will happen. Now, one point to note here that the same validation will take place at the server side as well. Suppose for any reason, user has disabled the "***Javascript***" in the browser, in that case model binder will do the same validation at the server side

and return the same view to the user to fix it.

Obviously, we could go ahead and write some custom validations if required in the project. Basically, we keep custom logic in the same model class which we are validating.

Summary:-

So, in this chapter we have added the CRUD operation. We have also seen Model Binding in action. We have covered EF Migrations which help us to test the app from sample data and also helps in keeping the model classes in

sync with database. We have also seen how to use different Scaffolding templates while generating codebase for CRUD operation. In the end, we have covered client side validation using Data Annotations.

Chapter 6: JavaScript and AJAX in MVC

WHAT DO you find in this CHAPTER?

⇒ Introduction

⇒ Managing Scripts

⇒ Ajax Helpers

⇒ Async Search

⇒ Paging

⇒ Summary

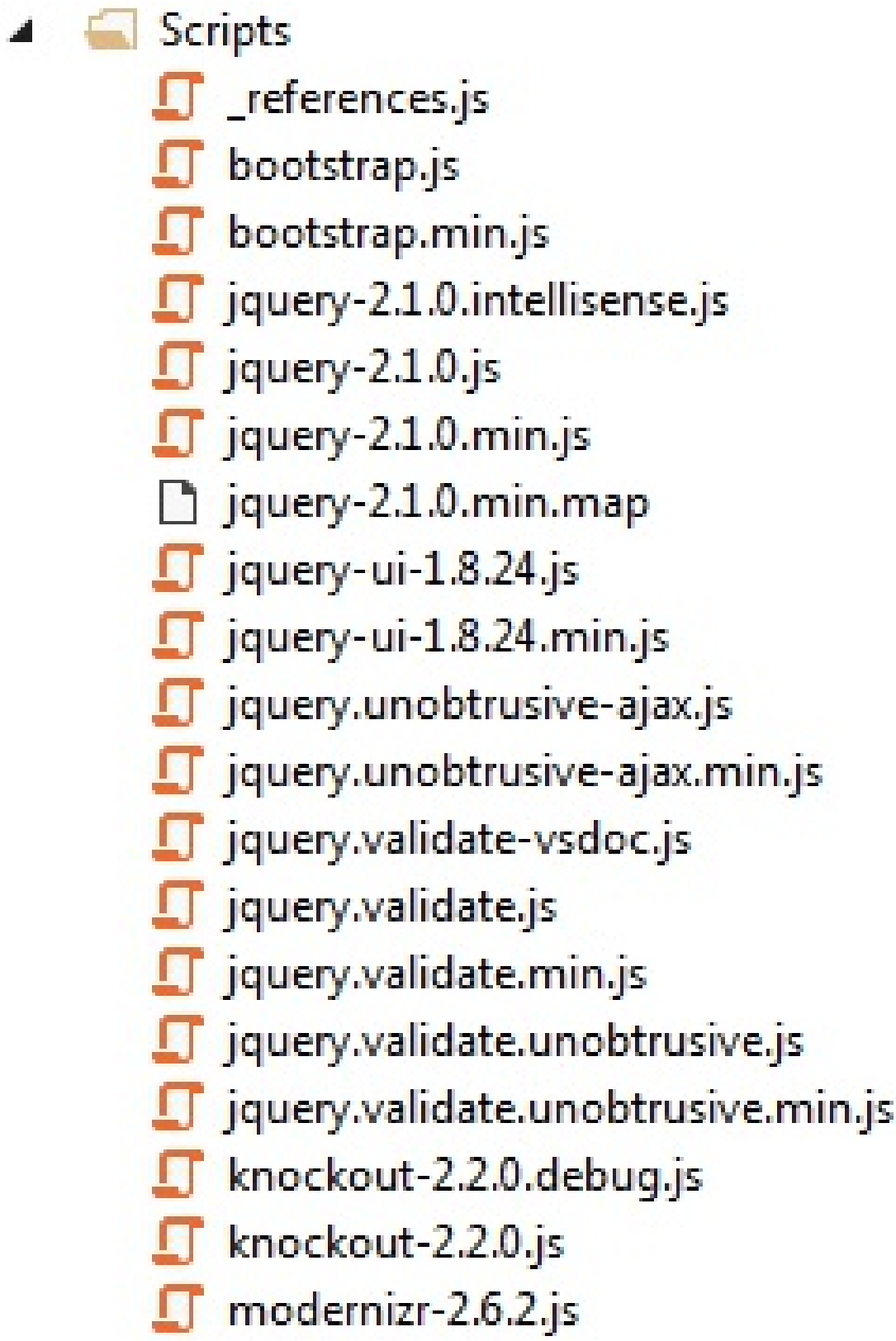
Introduction:-

Hi friends, in this section we'll look at adding client side features in the MVC application. So, in this entire chapter our main concern would be on adding AJAX features in MVC application. So, during the course we'll look at the Ajax helpers

offered by MVC. We'll also use Jquery and Jquery features to go to the various sections of the page and modify the same. In this section we'll stress upon how client validation works also how bundling and minification can be done. We'll also see how to produce and utilize JSON in the application.

Managing Scripts:-

In this case we'll first see that how Visual Studio organized the client side scripts. So, as you can see in the screen shot below that there is one scripts folder created by visual studio where in all the application related scripts reside.



It would be good idea to get started with client side development, by first

understanding what different kind of scripts provided by Microsoft by default. We'll start with the 1st one "***_reference.js***". This file usually gives Visual Studio a list of files which you commonly use so that you can get intellisense support while writing the javascript code. So, if we open the file and see that, it contains references of all those files which are required basically while writing the script.

```
/// <reference path="jquery-2.1.0.js" />
/// <reference path="jquery-ui-1.8.24.js" />
/// <reference path="jquery.validate.js" />
/// <reference path="jquery.validate.unobtrusive.js" />
/// <reference path="knockout-2.2.0.debug.js" />
/// <reference path="modernizr-2.6.2.js" />
```

So, we can see the list of different files which we usually use while writing our

client side code. If there is any other third party script you are using in your application, you could obviously go ahead and add the reference for the same in the file. After references, I have one third party **JS** that is "***bootstrap js***" which I have included to support the modified UI. After that there are different JQuery files. The one ending with intellisense is again to help you while writing the scripts. This file is responsible for providing much better intellisense support. The file "***jquery-2.1.0***" is the **Core JQuery library** file which we'll be using extensively for our client side coding. JQuery file ending with **.min** is a minified version of the JQuery core library file. Minification is

the process where in we are making the JavaScript file as small as possible by removing couple of redundant things like unneeded white space, unneeded comments, also by shortening the long function names.

But, when you open this file, you find that this file is not readable. Rather, this file is much smaller than the core version, means browser will take very less time to download the file when it is sent to the browser. So, these files we basically use in the production to boost the performance. Here, in this scripts folder, we also have "***Jquery-UI***" version of files which is meant for JQuery widgets like "***accordions,***

panels etc" which you can use in your application. We also have "***Jquery.Validate***", another plugin for Jquery meant for client side validations. So, the client side validations which we have discussed in the last chapter actually rely on this library.

We also have "***Knockout***" JS library which is meant for "***MVVM(Model-View-ViewModel)***" pattern. It provides features like two way data binding. Then we have script known as "***Modernizer***" which is a library basically to detect and enable HTML5 features on the browser. Other files which are equally important are "***Unobtrusive***" JS files. So, basically these files serve as a bridge

between ASP.Net MVC and Jquery. Like in client side validation in MVC you need both Jquery Validation Library, "***Jquery and Jquery.Validate.Unobtrusive***" script file as it takes metadata that's emitted by HTML Helper like "***EditorFor***" and passes that data to the Jquery validation such that it can be applied on the validation rules of the client. Libraries like Jquery and modernizer are typically required everywhere in the application, so it means it has to be included in the **Layout Page**.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>@ViewBag.Title - My ASP.NET MVC Application</title>
    <link href="~/favicon.ico" rel="shortcut icon" type="image/x-icon" />
    <meta name="viewport" content="width=device-width" />
```

```
    <link href="~/Content/bootstrap.min.css" rel="stylesheet" />
    <script src="~/Scripts/jquery-2.1.0.min.js"></script>
    <script src="~/Scripts/bootstrap.js"></script>
    <link href="~/Content/BootStrap-theme-movie.css" rel="stylesheet" />
    @*@Styles.Render("~/Content/css")*@
    @Scripts.Render("~/bundles/modernizr")
</head>

<div class="container">
    <div class="row">
        <div class="navbar navbar-default">
            <div class="navbar-header">
                <ul class="nav navbar-nav">
                    <li><span class="navbar-brand">Select One</span></li>

                </ul>
            </div>
            <div class="navbar-collapse navbar">
                <ul class="nav navbar-nav">
                    <li>@Html.ActionLink("Home", "Index", "Home")</li>
                    <li>@Html.ActionLink("About", "About", "Home")</li>
                    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
                    <li>@Html.ActionLink("Movies", "Index", "Movies")</li>
                    <li>@Html.Partial("_LoginPartial")</li>
                </ul>

            </div>
            @RenderSection("featured", required: false)

        </div>

    </div>
    <div>
        @RenderBody()

    </div>
    <footer>
        @*<div class="panel-footer">
            Best Review
            @Html.Action("bestReviews", "MovieReviews")
        </div>*@
    </footer>
</div>
        @Scripts.Render("~/bundles/jquery")
        @RenderSection("scripts", required: false)
```

```
</html>
```

So, as you can see at the top of the layout page, there is a call with "***Scripts.Render***" which invokes modernizer. So, what this "***Scripts.Render***" will do here, it will put a script tag referencing **Modernizer JS file** at the top of the markup inside the head tag. It is important to include Modernizer at the top of the page, because it starts executing before the rest of the HTML starts getting parsed.

Modernizer ensures that older browser like IE6 are good to work with HTML 5 elements like "***nav***" and "***section***" which you can see in the layout page.

However, other libraries are not required to be included at the top of the page; instead the same can be included at the bottom of the page as you can see above in the code snippet.

So, in this another call of the "***Scripts.Render***", it involves JQuery. It is always a good idea to include scripts at the bottom of the page, as it makes the page load faster. Now, let's understand the meaning of "***Scripts.Render***" or "***Styles.Render***". They both rely on the new features of ASP.Net which **bundles** and **minify** files for you.
"***Scripts.Render***" will give you minified Javascript bundle however,
"***Styles.Render***" will give you minified

bundle of **StyleSheet**. Now, another question which arises here, what's a bundle. Now, a bundle is yet another technique to boost the performance of a webpage by combining files together. So, browser doesn't need to download individual files, rather it will just download one file. But, during development time it is preferred to use the individual files, as you can see them and understand the code as it would be in readable format. Now, to understand how these bundles get initialized, let's open "***Global.asax***" file.

```
public class MvcApplication : System.Web.HttpApplication
  {
      protected void Application_Start()
      {
          AreaRegistration.RegisterAllAreas();

          WebApiConfig.Register(GlobalConfiguration.Configuration);
          FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
          RouteConfig.RegisterRoutes(RouteTable.Routes);
          BundleConfig.RegisterBundles(BundleTable.Bundles);
          AuthConfig.RegisterAuth();
      }
  }
```

In here, we can see that there is a call to "***BundleConfig***" to register the bundle. Now, when you press F12 on the "***BundleConfig***", it will take you to its definition which sits in the "***App_Start***" folder. Below is the simple class definition of "***BundleConfig***" where we can see that it's adding various scripts under scripts bundle and styles under styles bundle. We could obviously go

ahead and add as many scripts as we want in the existing bundles or create new custom bundles as well.

```
using System.Web;
using System.Web.Optimization;
```

```
namespace MovieReview
{
    public class BundleConfig
    {
        //      For      more      information      on      Bundling,      visit
http://go.microsoft.com/fwlink/?LinkId=254725
        public static void RegisterBundles(BundleCollection bundles)
        {
            bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
                        "~/Scripts/jquery-{version}.js"));

            bundles.Add(new ScriptBundle("~/bundles/jqueryui").Include(
                        "~/Scripts/jquery-ui-{version}.js"));

            bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(
                        "~/Scripts/jquery.unobtrusive*",
                        "~/Scripts/jquery.validate*"));

            // Use the development version of Modernizr to develop with and learn
from. Then, when you're
            // ready for production, use the build tool at http://modernizr.com to
pick only the tests you need.
            bundles.Add(new ScriptBundle("~/bundles/modernizr").Include(
                        "~/Scripts/modernizr-*"));

            bundles.Add(new
StyleBundle("~/Content/css").Include("~/Content/site.css"));

            bundles.Add(new StyleBundle("~/Content/themes/base/css").Include(
                        "~/Content/themes/base/jquery.ui.core.css",
                        "~/Content/themes/base/jquery.ui.resizable.css",
                        "~/Content/themes/base/jquery.ui.selectable.css",
                        "~/Content/themes/base/jquery.ui.accordion.css",
                        "~/Content/themes/base/jquery.ui.autocomplete.css",
                        "~/Content/themes/base/jquery.ui.button.css",
                        "~/Content/themes/base/jquery.ui.dialog.css",
                        "~/Content/themes/base/jquery.ui.slider.css",
                        "~/Content/themes/base/jquery.ui.tabs.css",
                        "~/Content/themes/base/jquery.ui.datepicker.css",
                        "~/Content/themes/base/jquery.ui.progressbar.css",
                        "~/Content/themes/base/jquery.ui.theme.css"));
        }
    }
}
```

So, as you can see in the snippet, 1st we are reaching the bundle by giving the virtual path then the "***file/files***" which you want to include in the bundle are listed with the call "***Include***". If you want to include multiple files under the same bundle, then files can be separated by "," operator. We can also use wild cards here, so like if you need to include multiple files starting with specific patterns, you could obviously specify the same like "*"~/Scripts/jquery.validate*"*". So, here "*" is specifying the wild card. So, all the files starting with "***jquery.validate***" and ending with anything will get included in the bundle.

Now, once we specified the bundles, we can make a call to them by either using "***Scripts.Render***" or "***Styles.Render***". These methods will behave differently depending upon your application configuration. Now, here it means how many files have been included in that bundle in the bundle.config, those many files will be bundled and produced at the client side. Let me explain the same by giving a small demo on that. On the Layout View just include this line "@Styles.Render("~/Content/themes/bas
So, this will include all the above listed files. Now save the file and refresh the page, now rt-click on the page and do view source, you will see 12 additional files have been included.

view-source:localhost:1033/Movies

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>Index - My ASP.NET MVC Application</title>
    <link href="/favicon.ico" rel="shortcut icon" type="image/x-icon" />
    <meta name="viewport" content="width=device-width" />
    <link href="/Content/bootstrap.min.css" rel="stylesheet" />
    <script src="/Scripts/jquery-2.1.0.min.js"></script>
    <script src="/Scripts/bootstrap.js"></script>
    <link href="/Content/BootStrap-theme-movie.css" rel="stylesheet" />

    <link href="/Content/themes/base/jquery.ui.core.css" rel="stylesheet"/>
<link href="/Content/themes/base/jquery.ui.resizable.css" rel="stylesheet"/>
<link href="/Content/themes/base/jquery.ui.selectable.css" rel="stylesheet"/>
<link href="/Content/themes/base/jquery.ui.accordion.css" rel="stylesheet"/>
<link href="/Content/themes/base/jquery.ui.autocomplete.css" rel="stylesheet"/>
<link href="/Content/themes/base/jquery.ui.button.css" rel="stylesheet"/>
<link href="/Content/themes/base/jquery.ui.dialog.css" rel="stylesheet"/>
<link href="/Content/themes/base/jquery.ui.slider.css" rel="stylesheet"/>
<link href="/Content/themes/base/jquery.ui.tabs.css" rel="stylesheet"/>
<link href="/Content/themes/base/jquery.ui.datepicker.css" rel="stylesheet"/>
<link href="/Content/themes/base/jquery.ui.progressbar.css" rel="stylesheet"/>
<link href="/Content/themes/base/jquery.ui.theme.css" rel="stylesheet"/>

    <script src="/Scripts/modernizr-2.6.2.js"></script>

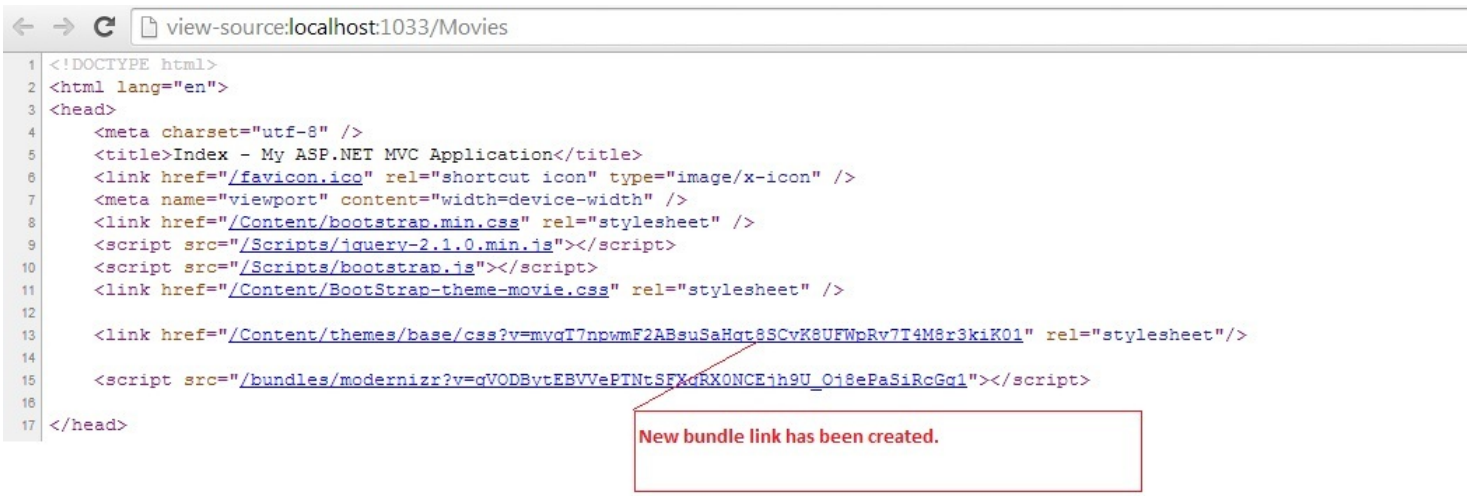
</head>

```

But, this is the exactly opposite behavior of bundling and minification. Bundling means combine all the scripts or styles in one file and produce the same at the client. To explain this scenario, since we are in debug mode, hence Visual Studio emitted all the files in original mode. Now, if I go ahead and change the debug settings from the web.config file to

```
<compilation debug="false"
targetFramework="4.5" />
```

Then, if I refresh the page and see the source of the file, it will produce the below result.



Now, when you try opening the link, in the browser then it will produce one file in the minified version. So, basically this file comprise of all the files in one bundle in minified version.



Now, let me go ahead undo the changes what I have done in Layout view, because I don't need that.

Ajax helpers:-

Now since we have seen that how to manage the scripts and use the same in our application, it's time to change the gear and switch to learn how Ajax helpers can be used. We'll start

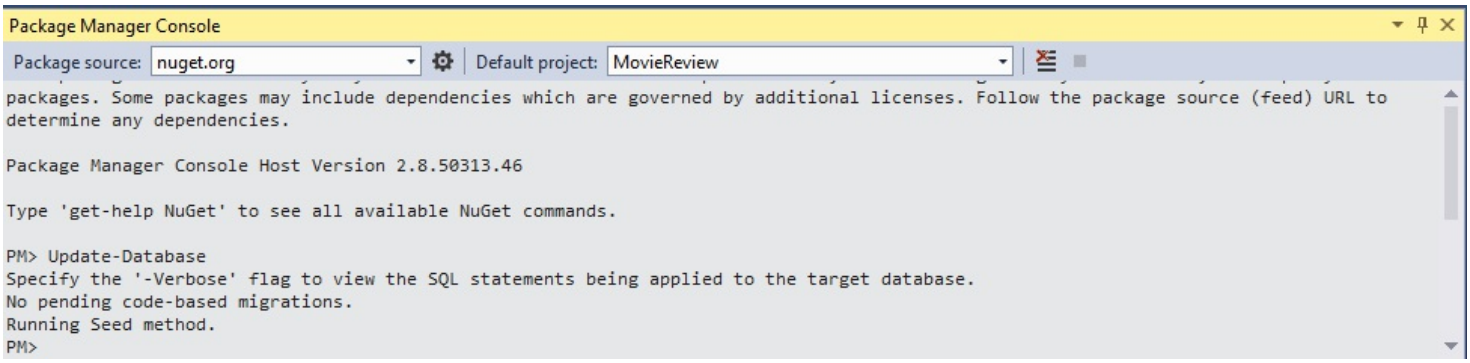
implementing the same from the home page itself. Now, let's do one quick change here, instead of using limited no of movies, just use 100 dummy movies, so that we can have some rich dataset to search for in the database. And the easiest way to achieve the same is seed the database with 100 odd movies. So, I'll go to the configurations file in the Migrations folder and change the script a bit. Now, it will look like as shown below:-

```
  protected override void Seed(MovieReview.Models.MovieDb context)
        {
            //  This method will be called after migrating to the latest version.

            context.Movies.AddOrUpdate(r => r.MovieName,
                new Movie { MovieName = "Avatar", DirectorName = "James Cameron",
ReleaseYear = "2009" },
                new Movie { MovieName = "Titanic", DirectorName = "James Cameron",
ReleaseYear = "1997" },
                new Movie { MovieName = "Die Another Day", DirectorName = "Lee
Tamahori", ReleaseYear = "2002" },
                new Movie
                {
                    MovieName = "Godzilla",
                    DirectorName = "Gareth Edwards",
                    ReleaseYear = "2014",
                    reviews = new List<MovieReviews>{
                       new
MovieReviews{ReviewerRating=5,ReviewerComments="Excellent",ReviewerName="Rahul
Sahay"}
                    }
                });
            for (int i = 0; i < 100; i++)
            {
                context.Movies.AddOrUpdate(m => m.MovieName,
                new   Movie   {   MovieName   =   "Anony_Movie_"   +   i.ToString(),
DirectorName = "Anony_Director_" + i.ToString(), ReleaseYear = "2014" });
            }
        }
```

So, as you can see in the above snippet, I have added one loop to add 100 odd movies in the database with anonymous movie name and director name. Now, with the above piece of code in place I

should be able to update the database, and for that I'll open my package manager console and run the update command ***(Update-Database)*** as shown below.



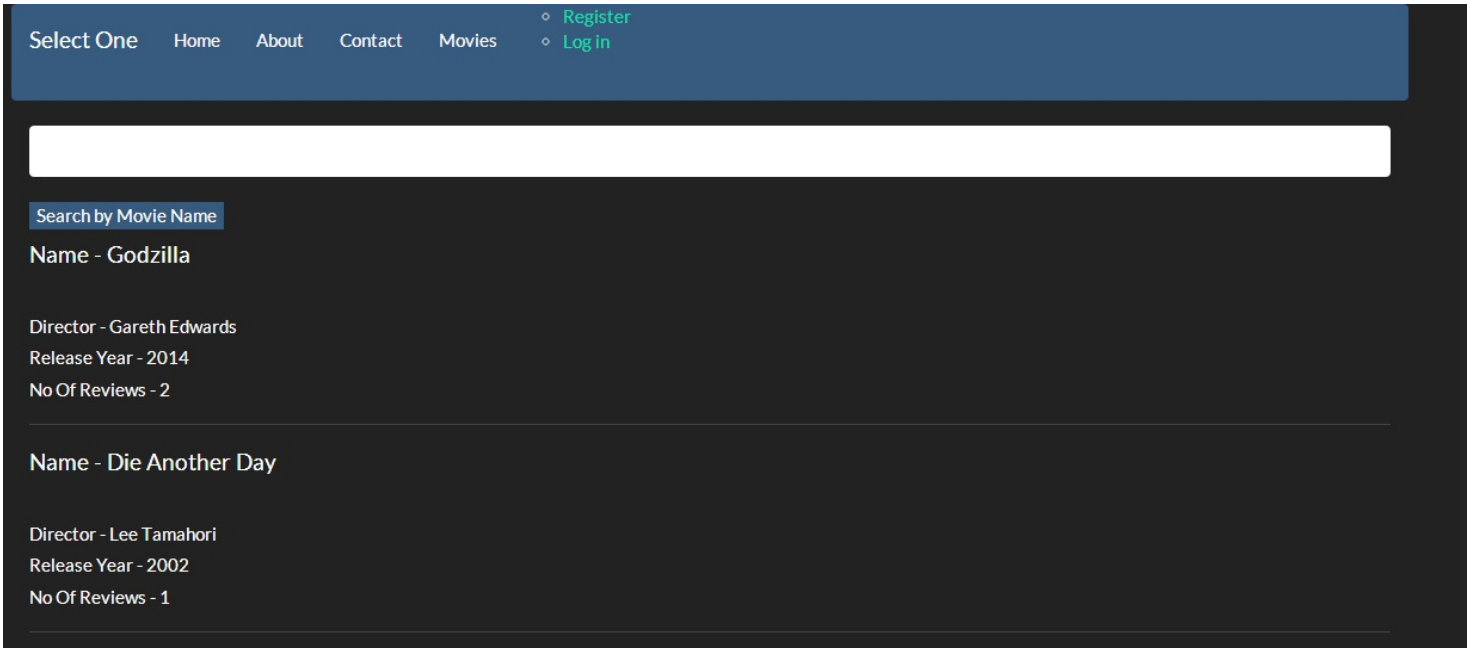
Now, as you can see in the above screen shot that seed method is completed successfully. Now, in order to see these new records on the home page, I also need to modify my LINQ query in the home controller. Initially it is designed to take 3 records. Now, I have modified the same to take 10 records as shown

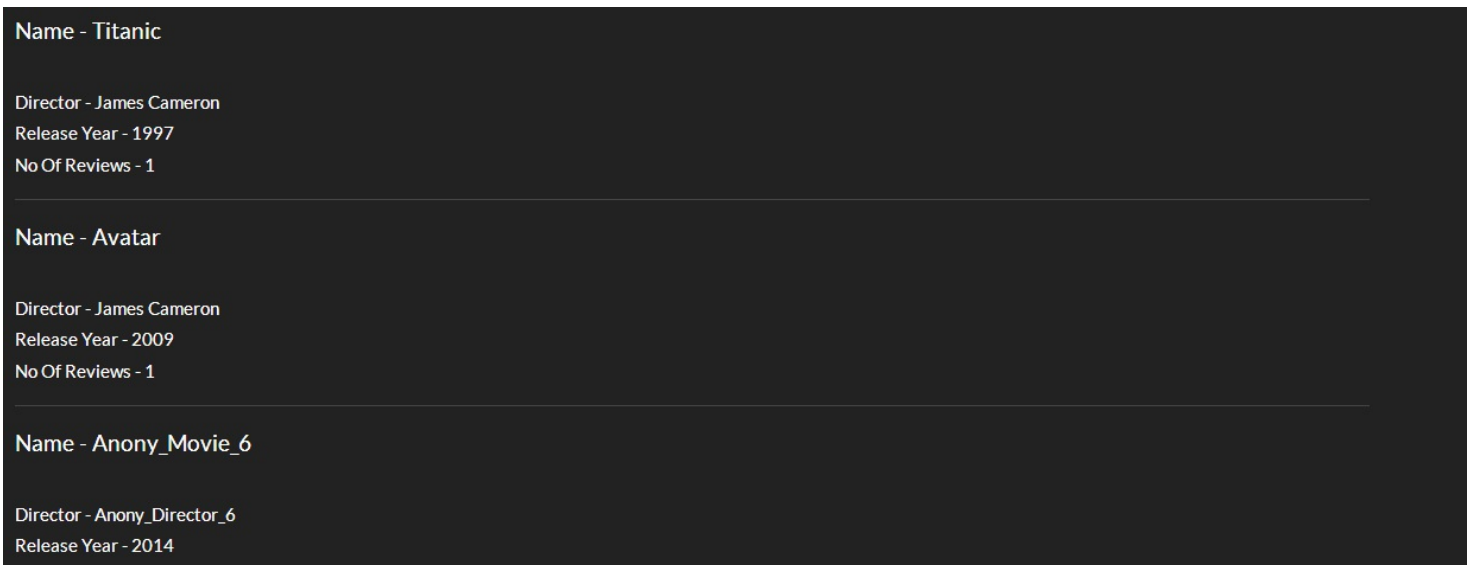
below.

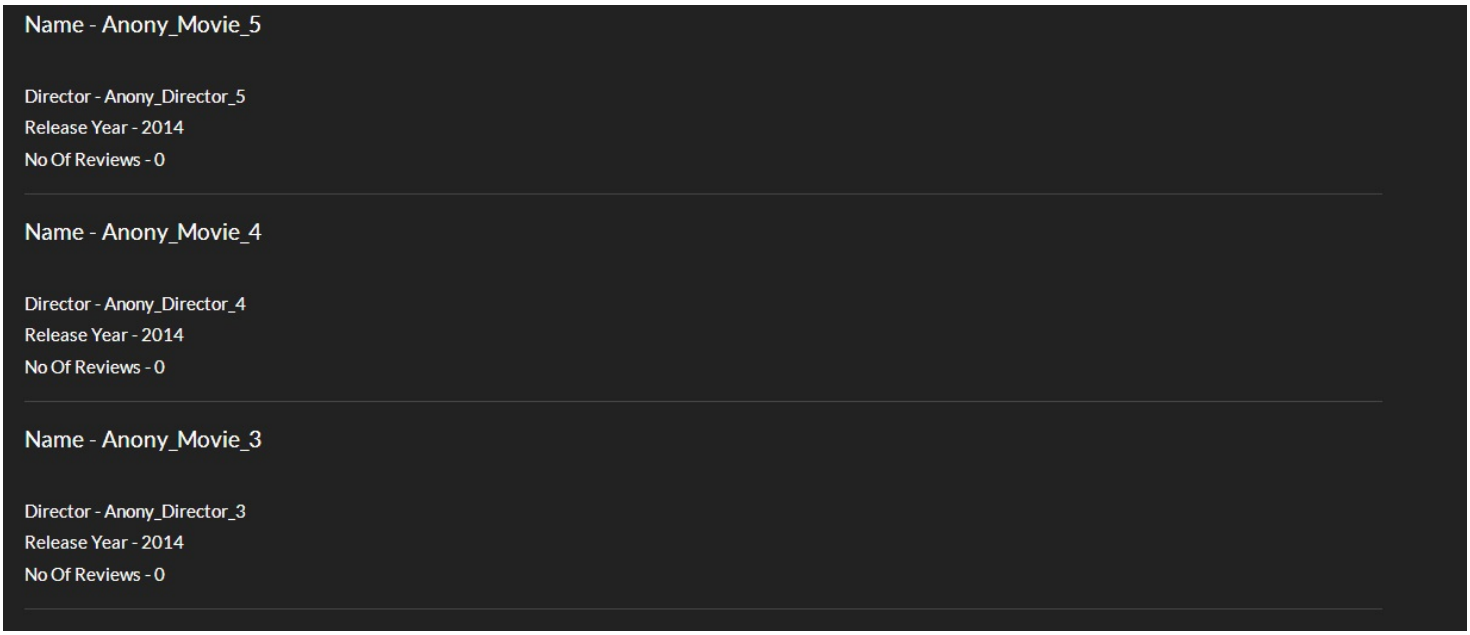
```
public ActionResult Index(string SearchQuery = null)
        {

            var model = _movieDb.Movies
                        .OrderByDescending(m => m.reviews.Count())
                        .Where(m      =>      SearchQuery      ==      null      ||
m.MovieName.StartsWith(SearchQuery))
                        .Take(10)
                        .Select(m => new MovieViewModel
                        {
                            Id = m.Id,
                            MovieName = m.MovieName,
                            DirectorName = m.DirectorName,
                            ReleaseYear = m.ReleaseYear,
                            NoOfReviews = m.reviews.Count()
                        });
            return View(model);
        }
```

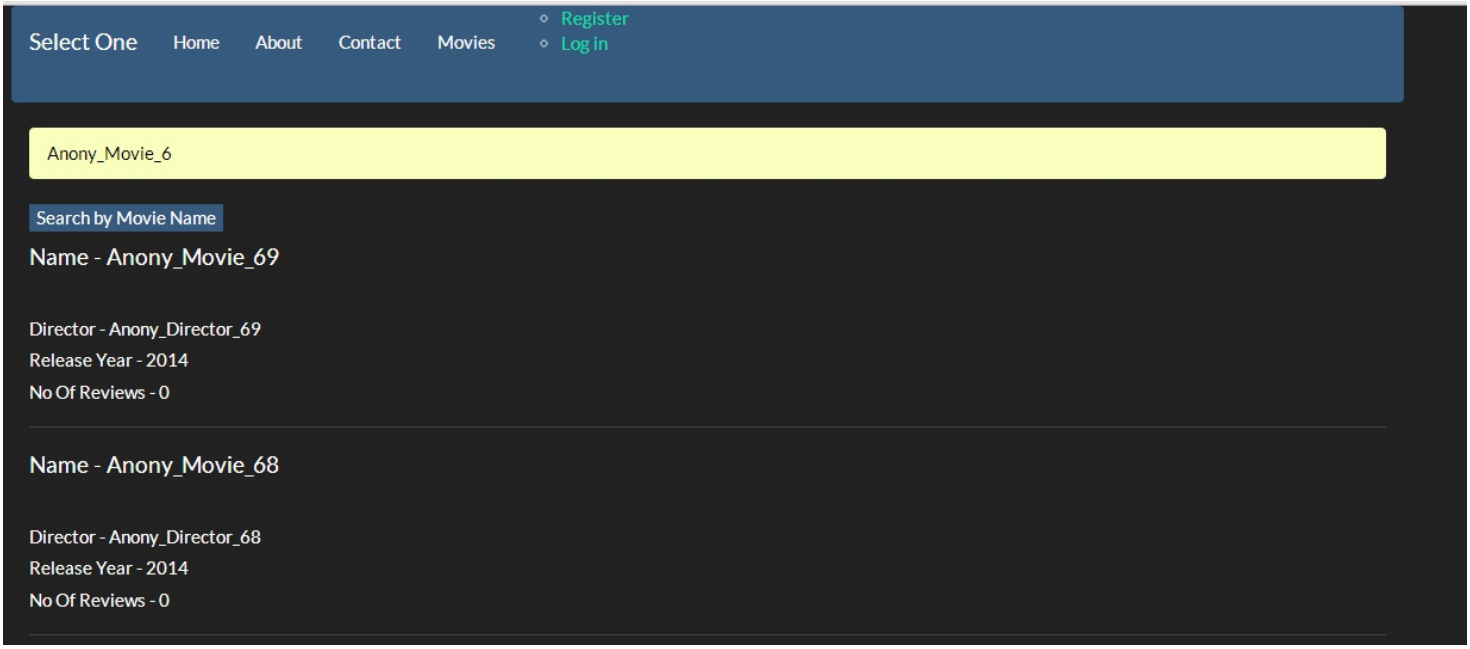
Now, with the above code in place I could go ahead build the app and refresh the home page, I should be able to see the newly inserted records in the database.


Select One
Home
About
Contact
Movies
Register
Log in
Search by Movie Name
Name - Godzilla
Director - Gareth Edwards
Release Year - 2014
No Of Reviews - 2
Name - Die Another Day
Director - Lee Tamahori
Release Year - 2002
No Of Reviews - 1


Name - Titanic
Director - James Cameron
Release Year - 1997
No Of Reviews - 1
Name - Avatar
Director - James Cameron
Release Year - 2009
No Of Reviews - 1
Name - Anony_Movie_6
Director - Anony_Director_6
Release Year - 2014



Likewise my database has been updated and same is getting reflected over here in the screen shot. Now, going ahead let's try searching for movie "***Anony_Movie_6***" as shown below in the screenshot.



Now, when I search the same it will produce me the above results. But, during that course it's redrawing the complete page, also losing the scroll position. Now, in order to fix the same I need to do the couple of things. 1st I need to create one new bundle which will be custom to my movie. So, in order to do that what I'll do, I'll go ahead and paste the below line of code in bundlesConfig file in the App_Start folder.

```
bundles.Add(new ScriptBundle("~/bundles/movies").Include(
                "~/Scripts/jquery-{version}.js",
                 "~/Scripts/jquery-ui-{version}.js",
                  "~/Scripts/jquery.unobtrusive*",
                  "~/Scripts/jquery.validate*"));
```

Now, as you can see in the above snippet I have taken all the dependent script under one "***movies***" bundle. Now, I need to search in the solution for "***@section Scripts {***", it will list me nine instances. Everywhere I need to remove "***@Scripts.Render("~/bundles/jqueryva*** and save the file. Then, I need to go to the Layout view and at the bottom of the page instead of loading ***"@Scripts.Render("~/bundles/jquery"***) we need to render ***"@Scripts.Render("~/bundles/movies"***

2nd thing I need to replace the index view snippet with the below snippet as shown below.

```
@model IEnumerable<MovieReview.Models.MovieViewModel>
@{
    ViewBag.Title = "Home Page";
}

@*<div>
    <form method="get">
        <input type="search" class="form-control has-success" name="SearchQuery">
        <br />


        <input type="submit" class="btn-primary" value="Search by Movie Name" />
    </form>
</div>*@

@using (Ajax.BeginForm(
    new AjaxOptions
    {
        HttpMethod = "get",
        InsertionMode = InsertionMode.Replace,
        UpdateTargetId = "movieDiv"
    }))
{
    <input type="search" class="form-control has-success" name="SearchQuery">
    <br />
    <input type="submit" class="btn-primary" value="Search by Movie Name" />
}

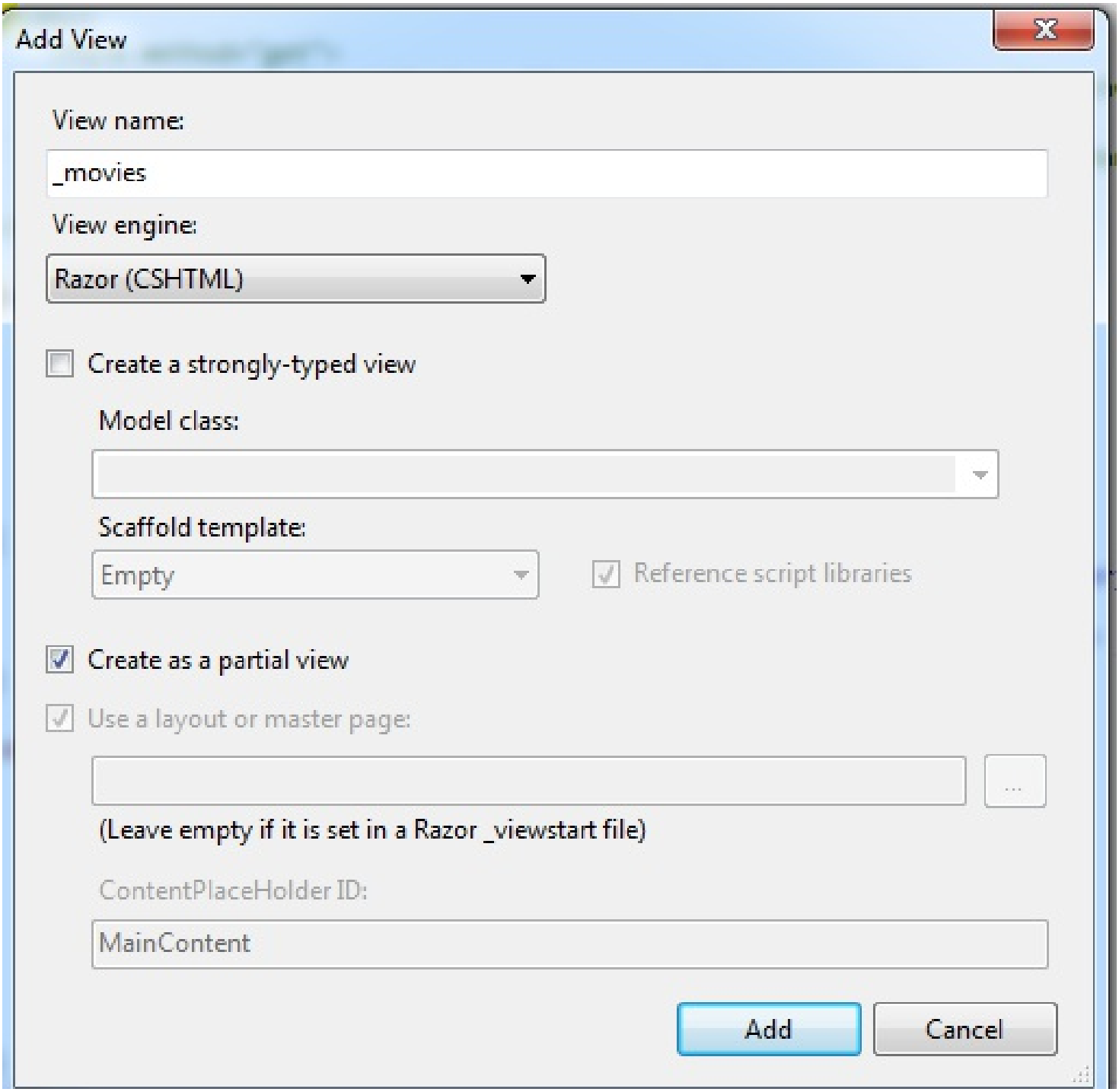
@Html.Partial("_movies",Model)
```

Now, in the above snippet as you can

see that I have used "***Ajax.BeginForm***", now Ajax.BeginForm is very similar to "***Html.BeginForm***". It writes a Form tag where in you can specify that what controllers and what route values need to be used. But, the major difference here is that it makes an asynchronous request to the server which draws only portion of the screen.

On the other hand, "***HTML.BeginForm***" makes a synchronous request to the server which redraws the entire screen. So, all we need to tell the helper that which portion of the page needs to be updated. And we can do this by the Ajax options object which we passed in.

So, what I have done here, I told the helper to make a "***get***" request to the server and whatever comes back should be replaced by the content in the page which is identified as "movieDiv". Also since I need to display the Movie List in two ways, one is the default way when the page is loaded and another one is when the movie is searched. So, it means we are going to need this list at multiple places. Hence, I just take out this div and create one partial view for that as shown below in the home folder.


Add View
View name:
_movies
View engine:
Razor (CSHTML)
Create a strongly-typed view
Model class:
Scaffold template:
Empty
Reference script libraries
Create as a partial view
Use a layout or master page:
...
(Leave empty if it is set in a Razor _viewstart file)
ContentPlaceHolder ID:
MainContent
Add
Cancel

```
@model IEnumerable<MovieReview.Models.MovieViewModel>


<div id="movieDiv">
    @foreach (var item in Model)
    {
        <div>
            <h4>Name - @item.MovieName</h4><br />
            <h5>Director - @item.DirectorName</h5>
            <h5>Release Year - @item.ReleaseYear</h5>
            <h5>No Of Reviews - @item.NoOfReviews</h5>
            <hr />
        </div>
    }
</div>
```

Now, it's entirely dependent on the controller action to decide to load the complete page or partial page. So, we also need to modify the home controller as shown below.

```
public ActionResult Index(string SearchQuery = null)
        {

            var model = _movieDb.Movies
                        .OrderByDescending(m => m.reviews.Count())
                        .Where(m=>SearchQuery==null||
m.MovieName.StartsWith(SearchQuery))
                        .Take(10)
                        .Select(m => new MovieViewModel
                        {
                            Id = m.Id,
                            MovieName = m.MovieName,
                            DirectorName = m.DirectorName,
                            ReleaseYear = m.ReleaseYear,
                            NoOfReviews = m.reviews.Count()
                        });
            if (Request.IsAjaxRequest())
            {
                return PartialView("_movies", model);
            }
            return View(model);
        }
```

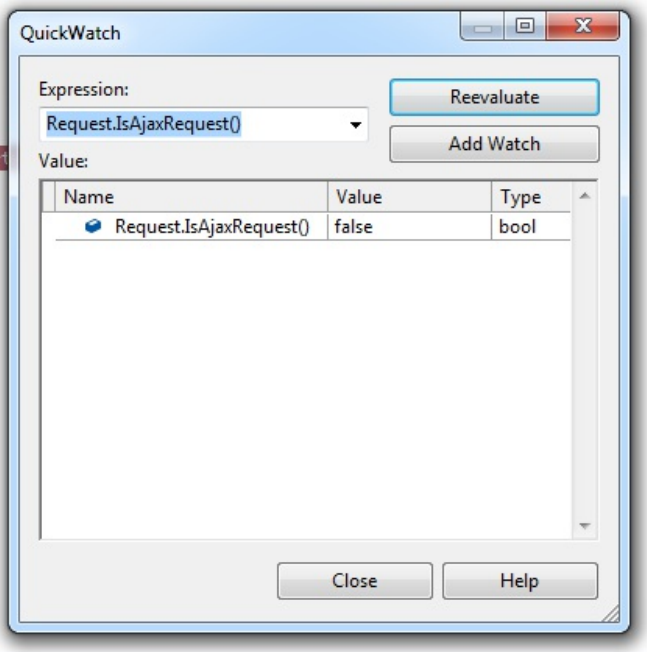
So, here what the above piece of code is checking; if the request arriving is the Ajax request or simple request. If it's an Ajax request, then it will simply return the partial view with the model value passed inside otherwise it will simply

return the plain view. Now, in order to prove my point, let's go ahead and put break point in the home controller and see how it behaves when the page loads and when the search is made.

```
public class HomeController : Controller
{
    MovieDb _movieDb = new MovieDb();
    1 reference | 0/1 passing
    public ActionResult Index(string SearchQuery = null)
    {

        var model = _movieDb.Movies
                    .OrderByDescending(m => m.reviews.Count())
                    .Where(m => SearchQuery == null || m.MovieName.Start
                    .Take(10)
                    .Select(m => new MovieViewModel
                    {
                        Id = m.Id,
                        MovieName = m.MovieName,
                        DirectorName = m.DirectorName,
                        ReleaseYear = m.ReleaseYear,
                        NoOfReviews = m.reviews.Count()
                    });
        if (Request.IsAjaxRequest())
        {
            return PartialView("_movies", model);
        }
        return View(model);
    }

    1 reference | 0/1 passing
    public ActionResult About()
    {
        ViewBag.Message = "Your app description page.";
```

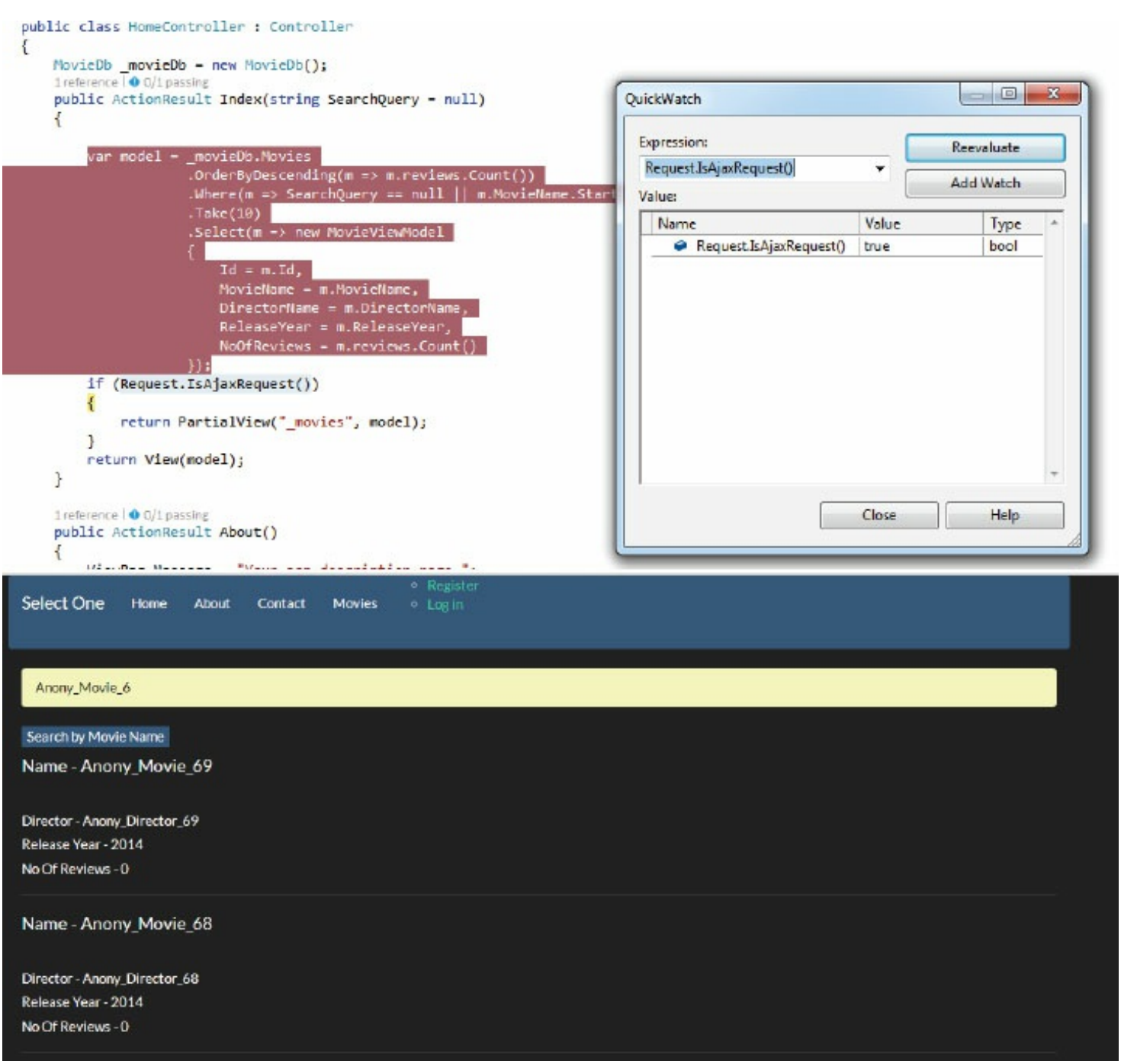


Above screen shot is the scenario where in page is loading, hence "Request.IsAjaxRequest()" returned me false. Now, let's go ahead and make a

search



Now, when I see the debugger, it will return me true as result and hence render me the partial view directly.



Async Search:-

Now, let's go ahead and make the search

an **Async Search**. Now, before jumping into action, let's check what are other Ajax helpers which could be useful in this case. There are basically three AJAX related features which MVC provided. They are:-

@Ajax.BeginForm:- Just now we have seen this; Ajax.BeginForm writes out a form tag and makes a request to the server asynchronously.

@Ajax.ActionLink:- Just like Ajax.BeginForm, Ajax.ActionLink creates a link to make an async request and update the screen instead of navigating the browser to a new page or new tab.

Client Side Validation:- The third approach is the JavaScript validation. All of these approach used in JavaScript validation which is known as Unobtrusive Javascript. We have seen this also before. Unobtrusive javascript means we don't have JS spread throughout in the view. Now, let's understand what happens behind the scene when search is made, in this case it is taken care by Ajax.BeginForm.

```
<form action="/" data-ajax="true" data-ajax-method="get" data-ajax-mode="replace" data-ajax-update="#movieDiv" id="form0" method="post">    <input type="search"
class="form-control has-success" name="SearchQuery">
    <br />
    <input type="submit" class="btn-primary" value="Search by Movie Name" />
</form>

<div id="movieDiv">
        <div>
            <h4>Name - Godzilla</h4><br />
            <h5>Director - Gareth Edwards</h5>
            <h5>Release Year - 2014</h5>
            <h5>No Of Reviews - 2</h5>
            <hr />
        </div>
        <div>
            <h4>Name - Die Another Day</h4><br />
            <h5>Director - Lee Tamahori</h5>
            <h5>Release Year - 2002</h5>
            <h5>No Of Reviews - 1</h5>
            <hr />
        </div>
        <div>
            <h4>Name - Titanic</h4><br />
            <h5>Director - James Cameron</h5>
            <h5>Release Year - 1997</h5>
            <h5>No Of Reviews - 1</h5>
            <hr />
        </div>
        <div>
            <h4>Name - Avatar</h4><br />
            <h5>Director - James Cameron</h5>
            <h5>Release Year - 2009</h5>
            <h5>No Of Reviews - 1</h5>
            <hr />
```

So, when you look at the source code, you can see that Ajax.BeginForm emitted data- (***known as datadash attribute***) attributes in the form tag. Now, **data-** attributes are part of the HTML 5 specifications. However, you can also write your own data- attribute for consuming the same in the browser. We'll see the same in some time.

Then, the scripts which we write, there will be some JavaScript which will go and interpret these data- attributes and start attaching the events to add Ajax features. Client side validation also works the same way. Let's open a review page, and then open it source code.

```
    <fieldset>
        <legend>MovieReviews</legend>

        <div class="form-group">
            <input data-val="true" data-val-number="The field Id must be a number." data-val-required="The Id field is required." id="Id" name="Id" type="hidden"
value="1" />
            <label for="ReviewerName">ReviewerName</label>
            <div class="form-control">
                    <input class="text-box single-line" id="ReviewerName" name="ReviewerName" type="text" value="Rahul Sahay" />
                    <span class="field-validation-valid" data-valmsg-for="ReviewerName" data-valmsg-replace="true"></span>
                </div>
            </div>
        <div class="form-group">
            <label for="ReviewerComments">ReviewerComments</label>

            <div class="form-control">
                <input class="text-box single-line" data-val="true" data-val-length="The field ReviewerComments must be a string with a maximum length of 200."
data-val-length-max="200" data-val-required="The ReviewerComments field is required." id="ReviewerComments" name="ReviewerComments" type="text" value="Excellent"
/>
                <span class="field-validation-valid" data-valmsg-for="ReviewerComments" data-valmsg-replace="true"></span>
            </div>
        </div>

        <div class="form-group">
            <label for="ReviewerRating">ReviewerRating</label>

            <div class="form-control">
                <input class="text-box single-line" data-val="true" data-val-number="The field ReviewerRating must be a number." data-val-range="The field
ReviewerRating must be between 1 and 5." data-val-range-max="5" data-val-range-min="1" data-val-required="The ReviewerRating field is required."
id="ReviewerRating" name="ReviewerRating" type="number" value="5" />
                <span class="field-validation-valid" data-valmsg-for="ReviewerRating" data-valmsg-replace="true"></span>
            </div>
        </div>
```

Here, also we can see data- attribute. Like if you see the Comments section, here in you can see that it is setting maximum value to 200, the error message to display when it gets exceeded and the required field as well. So, the JavaScript code coming here is finding these data- attributes, interpreting the same and then returning the result on the page.

It will also work when JavaScript

disabled, obviously, in that case we won't have client side validation. But it will work and nothing will break. So, in that case data- attributes won't be utilized, rather form will get posted and it will have the server side validation.

Now, let me customize the same by using our own JS code. Now, let's modify the index page as shown below.

```
@model IEnumerable<MovieReview.Models.MovieViewModel>
@{
    ViewBag.Title = "Home Page";
}

<div>
    <form method="get" action="@Url.Action("Index")" data-movie-ajax="true"
movie-target="#movieDiv">
        <input type="search" class="form-control has-success" name="SearchQu
        <br />
        <input type="submit" class="btn-primary" value="Search by Movie Name
    </form>
</div>

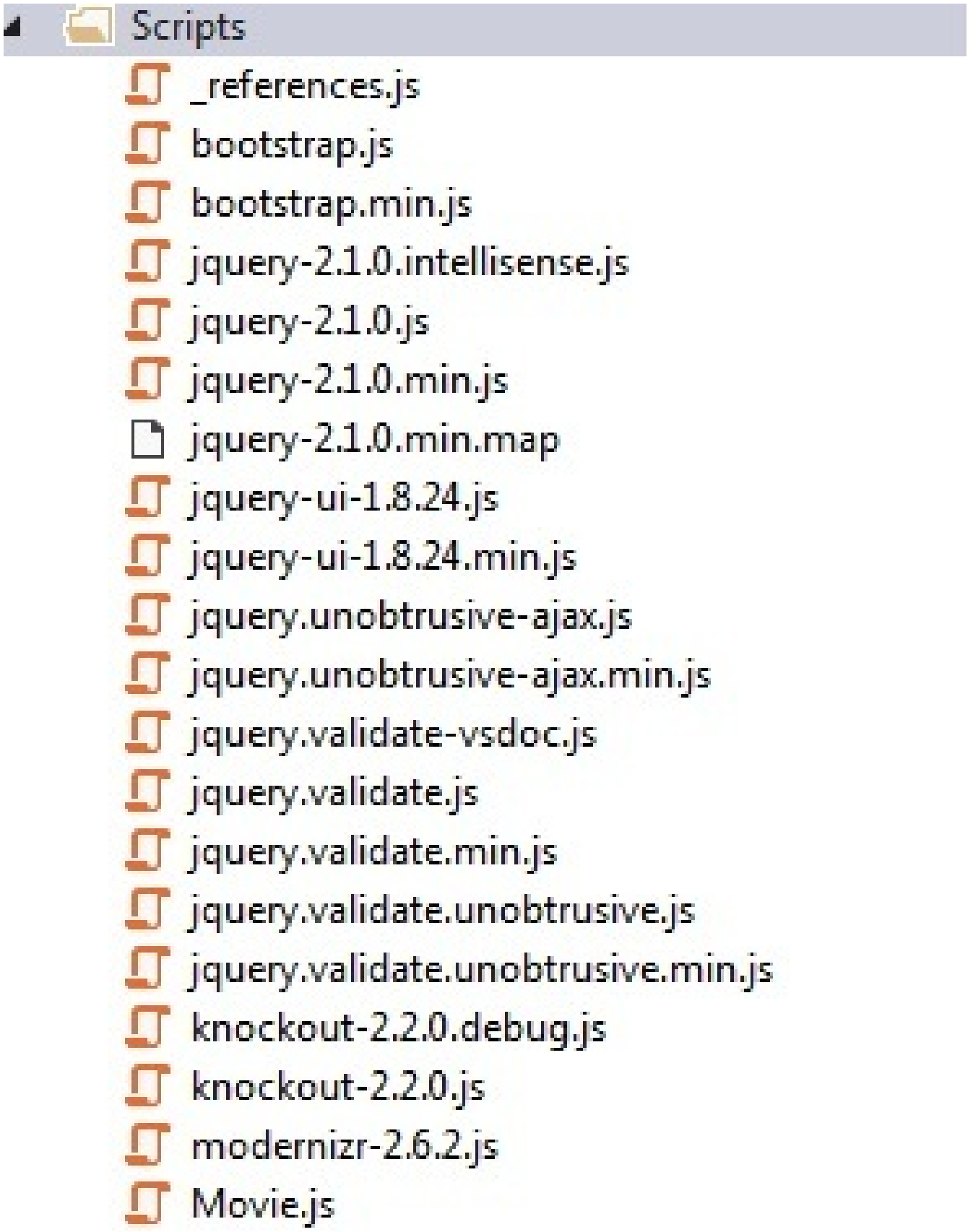
@*@using (Ajax.BeginForm(
```

```
    new AjaxOptions
    {
        HttpMethod = "get",
        InsertionMode = InsertionMode.Replace,
        UpdateTargetId = "movieDiv"
    }))
{
    <input type="search" class="form-control has-success" name="SearchQuery">
    <br />
    <input type="submit" class="btn-primary" value="Search by Movie Name" />
}*@

@Html.Partial("_movies",Model)
```

Here, I have just commented that Ajax.BeginForm Section. Rather, than that I have enabled the previous one by adding some additional piece of information. Now, here I have explicitly specified the action attribute to make sure that form has an action attribute using the URL helper to generate a URL controller action. Then, I have also added two **data-** attributes, one to identify that form should be **Ajax** and the

other one is specifying the **ID** of the div which needs to be updated. The idea behind this approach is using our own JavaScript code to fire and wire up the events behind the scene. Now, we need to add one JavaScript file in the scripts folder. So, I add the same with the name "***Movie.js***" as shown below.



Now, I need to include this file in my

bundle as I would need this script to be available everywhere.

```
bundles.Add(new ScriptBundle("~/bundles/movies").Include(
                    "~/Scripts/jquery-{version}.js",
                     "~/Scripts/jquery-ui-{version}.js",
                      "~/Scripts/jquery.unobtrusive*",
                      "~/Scripts/jquery.validate*",
                      "~/Scripts/Movie.js"));
```

Now, I need to put some code in there to invoke the search and return the result back to the screen. Below is the script in finished form.

```
$(function () {

    var MovieFormSubmit = function () {
        //Grab the refernce of the form
        var $form = $(this);

        //Build the options object
        var options = {
            url: $form.attr("action"),
            type: $form.attr("method"),
            data: $form.serialize()
        };

        $.ajax(options).done(function (data) {
            var target = $($form.attr("data-movie-target"));
            target.replaceWith(data);
        });

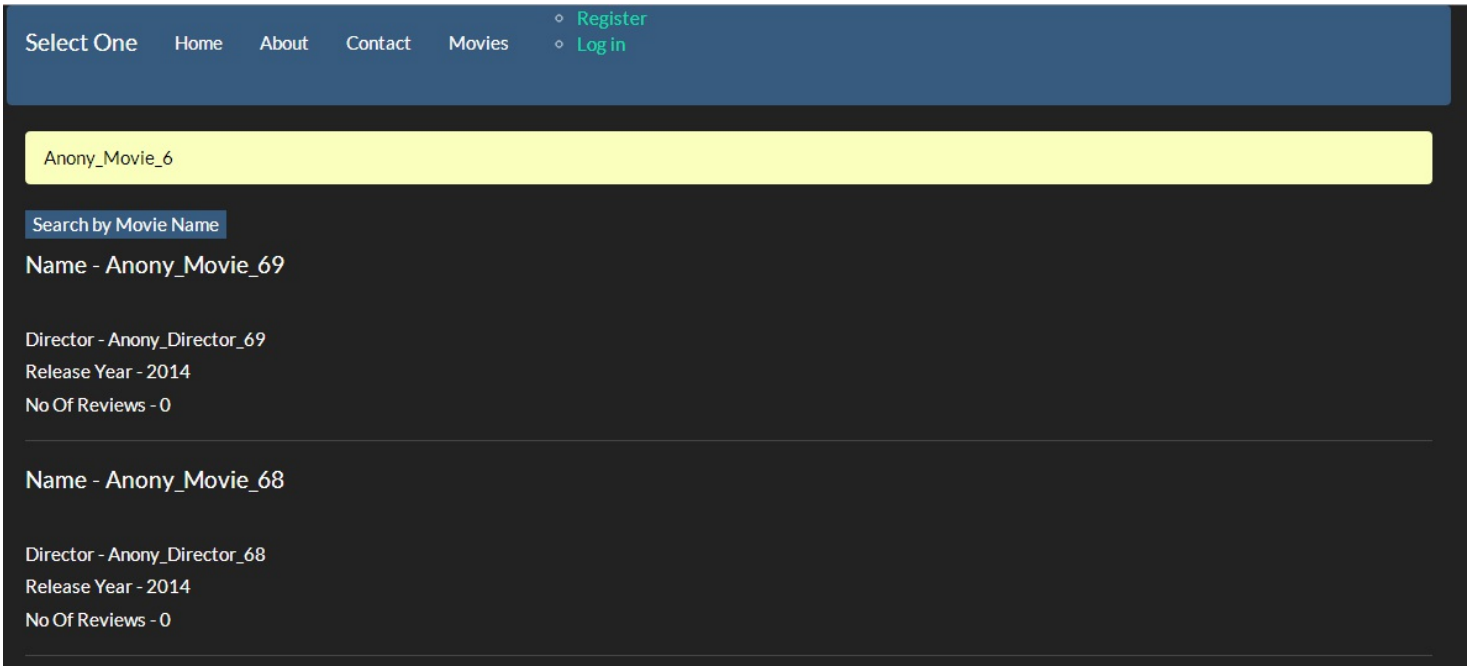
        //To prevent the browser from doing it's defualt action means navigating
away and redrawing the complete page
        return false;
    };
    //Look for Form with the name "data-movie-ajax", then wire up the submit event.
     $("form[data-movie-ajax='true']").submit(MovieFormSubmit);
});
```

Now, let me go ahead and explain the same step by step. So, here "***$(function({});***" is basically invoking the JQuery function. Then, I have used the JQuery selector to go out and find the form that has "***data-movie-ajax***" value set to **true**; then am wiring up the event

with "***submit***" event. So, when user clicks on the search, instead of posting back to the server, it will land up here in my JavaScript function; in this case it will land up in "***MovieFormSubmit***". So, inside this function I have written the code to handle the **Form submission**. So, here it's collecting all the parameters, sending them to the server, get the result back from the server and then send the same back to the browser.

Now, from the options object, I can get the **URL** by looking at the action attribute in the Form. Also, I have specified the type of the request which will again be fetched from the Form value itself. And finally, we need to

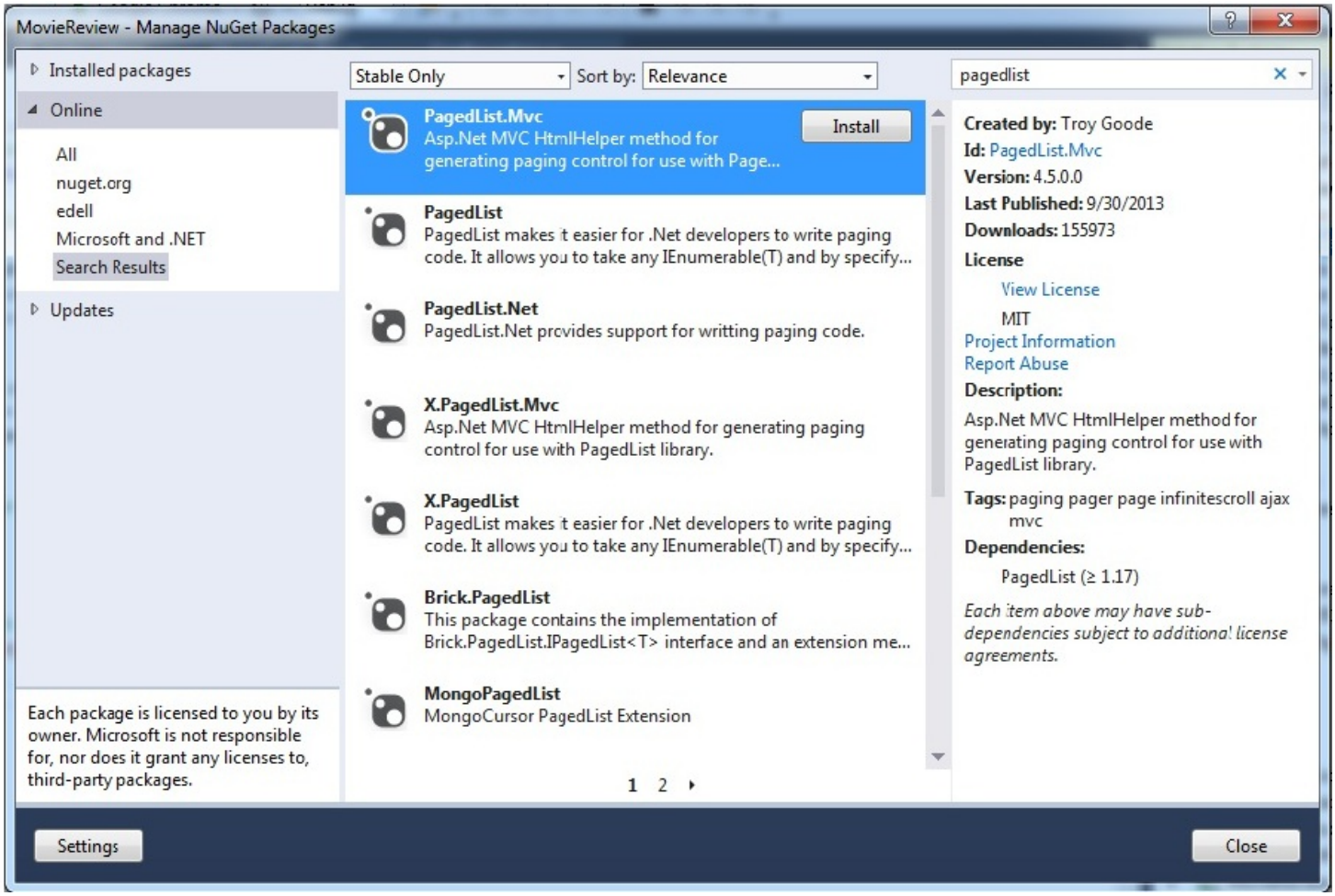
serialize the data which we are sending back to the server. Then, I have made the **Async** call by using "***$.ajax***" by passing the options object in there and when it is done, this function will come into action and this will have the response from the server in the data object. So, what I have done here, I go out and find the target with the div id "movieDiv". Now, let's go ahead build the app and test the searching facility once again. This time this should return the desired result with our own custom query.



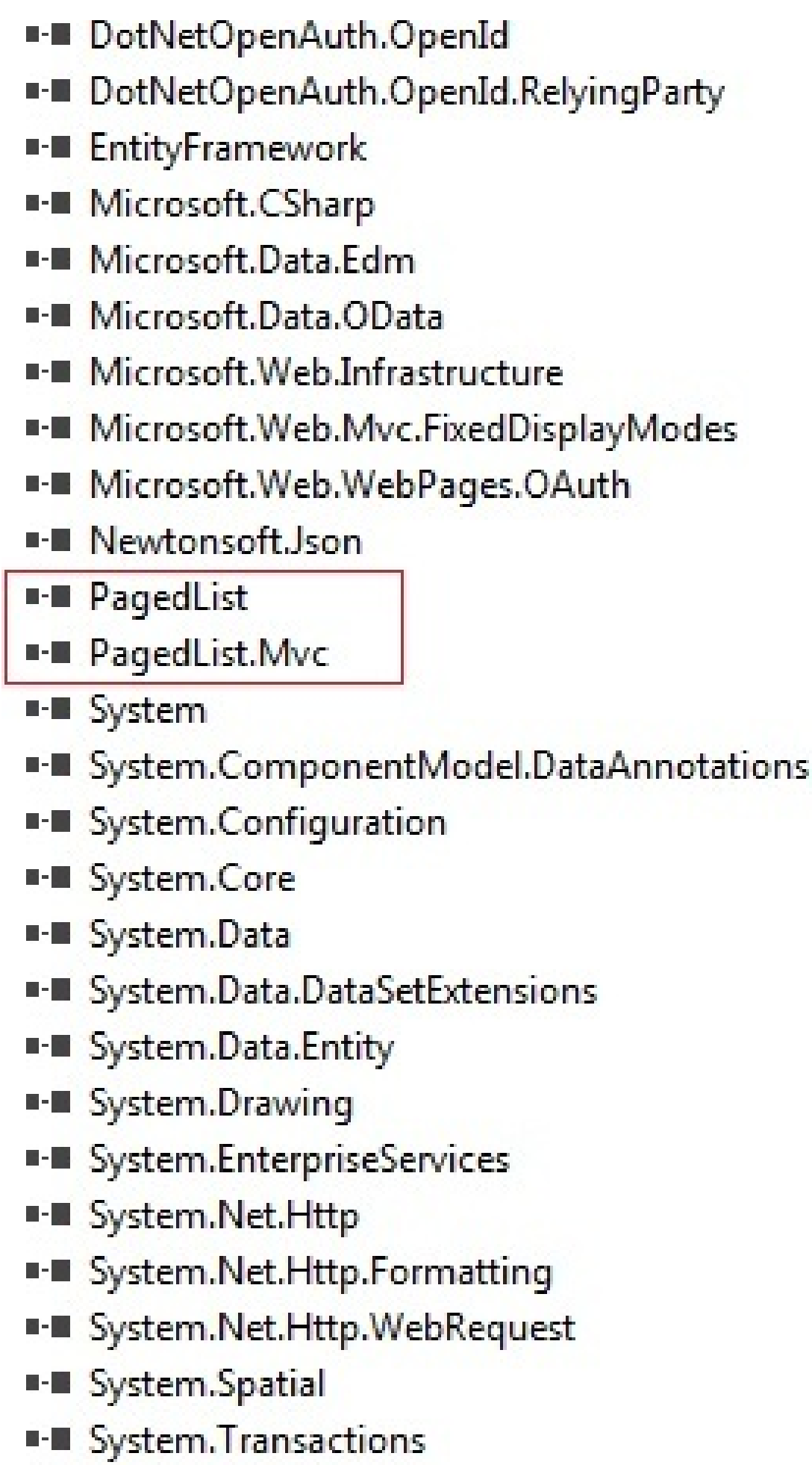
Paging:-

Now, finally I would like to add one more feature in this and that is paging. I would like to give an option to user so that they can page through in the result set. Now, to do the paging I need one third party component installed in application. And obviously to install third party component in our application,

best way is to use **NuGet Package Manager** as shown below.



Now, I would be installing the highlighted one "***PagedList.MVC***" in my project. Now, upon installation, we can verify the same in our application as shown below.


DotNetOpenAuth.OpenId
DotNetOpenAuth.OpenId.RelyingParty
EntityFramework
Microsoft.CSharp
Microsoft.Data.Edm
Microsoft.Data.OData
Microsoft.Web.Infrastructure
Microsoft.Web.Mvc.FixedDisplayModes
Microsoft.Web.WebPages.OAuth
Newtonsoft.Json
PagedList
PagedList.Mvc
System
System.ComponentModel.DataAnnotations
System.Configuration
System.Core
System.Data
System.Data.DataSetExtensions
System.Data.Entity
System.Drawing
System.EnterpriseServices
System.Net.Http
System.Net.Http.Formatting
System.Net.Http.WebRequest
System.Spatial
System.Transactions

Now, I can go ahead and start doing changes in our application. We'll start doing the same from our home controller. Now, the index action of our home controller will not be returning just 10 records, instead it would be returning page of records. Now, **pagedList** package will give me some helpers to setup the paging.

So, now in order to use these extension methods I need to include the namespace for the same as “using PagedList;”. Here, I'll be using “***ToPagedList***” as shown below in the index action. Now, this extension will take page no as declared in the method as a parameter.

This extension method will also take page size, So I'll specify 10 records per page. Now below is the modified indexation in its finished form.

```
1 reference | 0/1 passing
public ActionResult Index(string SearchQuery = null,int page=1)
{

    var model = _movieDb.Movies
                .OrderByDescending(m => m.reviews.Count())
                .Where(m => SearchQuery == null || m.MovieName.StartsWith(SearchQuery))
                .Se
                {
                    ToArray<>
                    ToDictionary<>
                    ToList<>
                    ToLookup<>
                    ToPagedList<>
                    ToString
                }).to;
    if (Request.IsAjaxRequest())
    {
        return PartialView("_movies", model);
    }
    return View(model);
}
```

(extension) IPagedList<T> IQueryable<T>.ToPagedList<T>(int pageNumber, int pageSize) (+ 1 overload(s))
(The documentation cache is still being constructed. Please try again in a few seconds.)

```
public ActionResult Index(string SearchQuery = null,int page=1)
        {

            var model = _movieDb.Movies
                        .OrderByDescending(m => m.reviews.Count())
                        .Where(m=>SearchQuery==null||m.MovieName.StartsWith(Search
Query))
                        .Select(m => new MovieViewModel
                        {
                            Id = m.Id,
                            MovieName = m.MovieName,
                            DirectorName = m.DirectorName,
                            ReleaseYear = m.ReleaseYear,
                            NoOfReviews = m.reviews.Count()
                        }).ToPagedList(page,10);
            if (Request.IsAjaxRequest())
            {
                return PartialView("_movies", model);
            }
            return View(model);
        }
```

That's it. This much was the change in server side. Now, we need to incorporate the changes for the same in the UI. So, obviously the 1st change in the view is to change the model type; as now the model type would be "***IPagedList***". Now, in order to resolve these new types, I need to update the

namespace in the "***web.config***" inside the "***Views***" folder as shown below.

```
<system.web.webPages.razor>

    <host    factoryType="System.Web.Mvc.MvcWebRazorHostFactory,    System.Web.Mvc,
Version=4.0.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" />
    <pages pageBaseType="System.Web.Mvc.WebViewPage">
      <namespaces>
        <add namespace="System.Web.Mvc" />
        <add namespace="System.Web.Mvc.Ajax" />
        <add namespace="System.Web.Mvc.Html" />
        <add namespace="System.Web.Optimization"/>
        <add namespace="System.Web.Routing" />
        <add namespace="PagedList"/>
        <add namespace="PagedList.Mvc"/>
      </namespaces>
    </pages>
  </system.web.webPages.razor>
```

So, here I have added two namespaces as shown above for the **pagedList**. Now, in order to take this effect, what I need to do is just close the Visual Studio and start again, and then it will take the effect.

```
@model IPagedList<MovieReview.Models.MovieViewModel>
@{
    ViewBag.Title = "Home Page";
}

<div>
    <form method="get" action="@Url.Action("Index")" data-movie-ajax="true" data-
movie-target="#movieDiv">
        <input type="search" class="form-control has-success" name="SearchQuery">
        <br />
        <input type="submit" class="btn-primary" value="Search by Movie Name" />
    </form>
</div>

@*@using (Ajax.BeginForm(
    new AjaxOptions
    {
        HttpMethod = "get",
        InsertionMode = InsertionMode.Replace,
        UpdateTargetId = "movieDiv"
    }))
{
    <input type="search" class="form-control has-success" name="SearchQuery">
    <br />
    <input type="submit" class="btn-primary" value="Search by Movie Name" />
}*@
```

```
@Html.Partial("_movies",Model)
```

Now, from this index view I am also rendering this partial view. So, I also need to change the model directive there in the partial view as well.

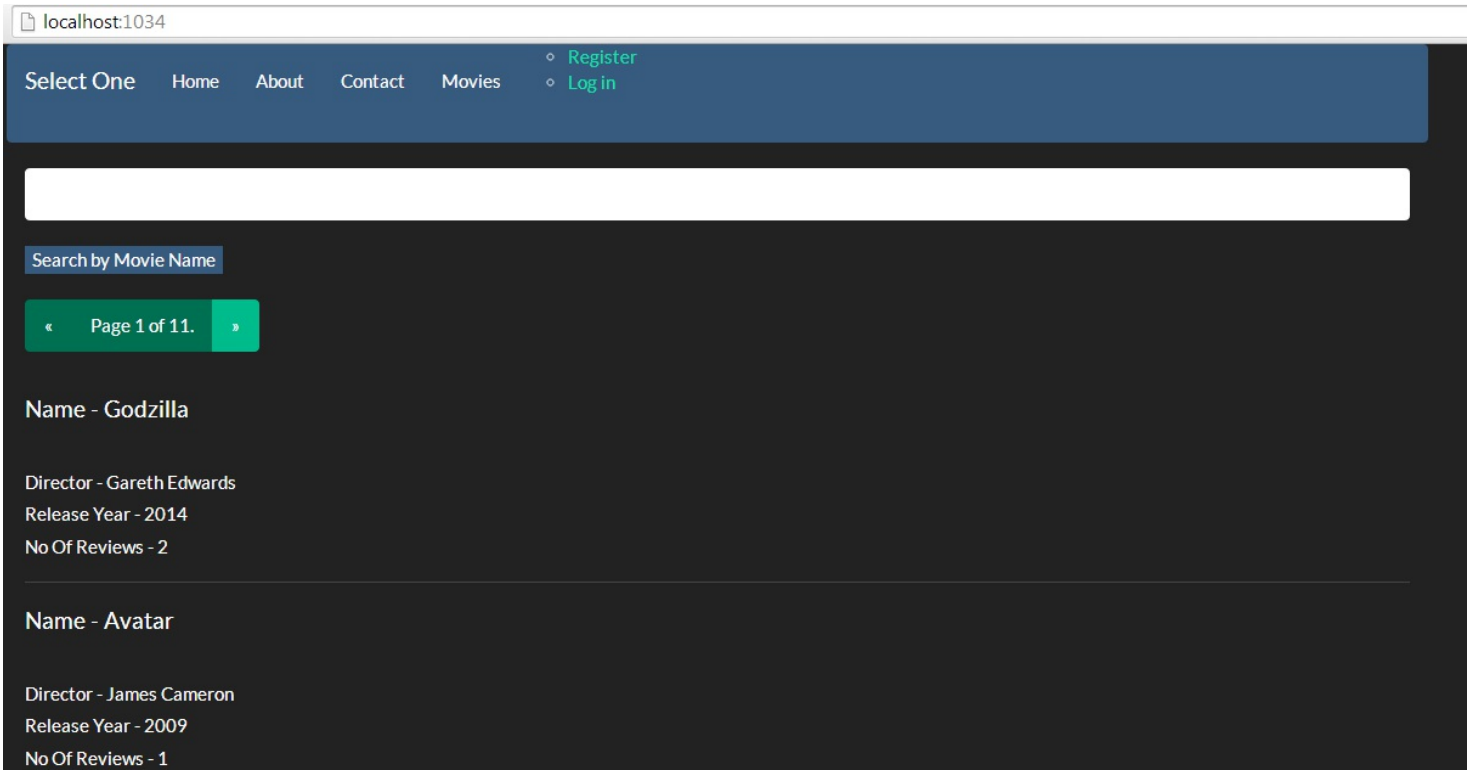
```
@model IPagedList<MovieReview.Models.MovieViewModel>

<div id="movieDiv">

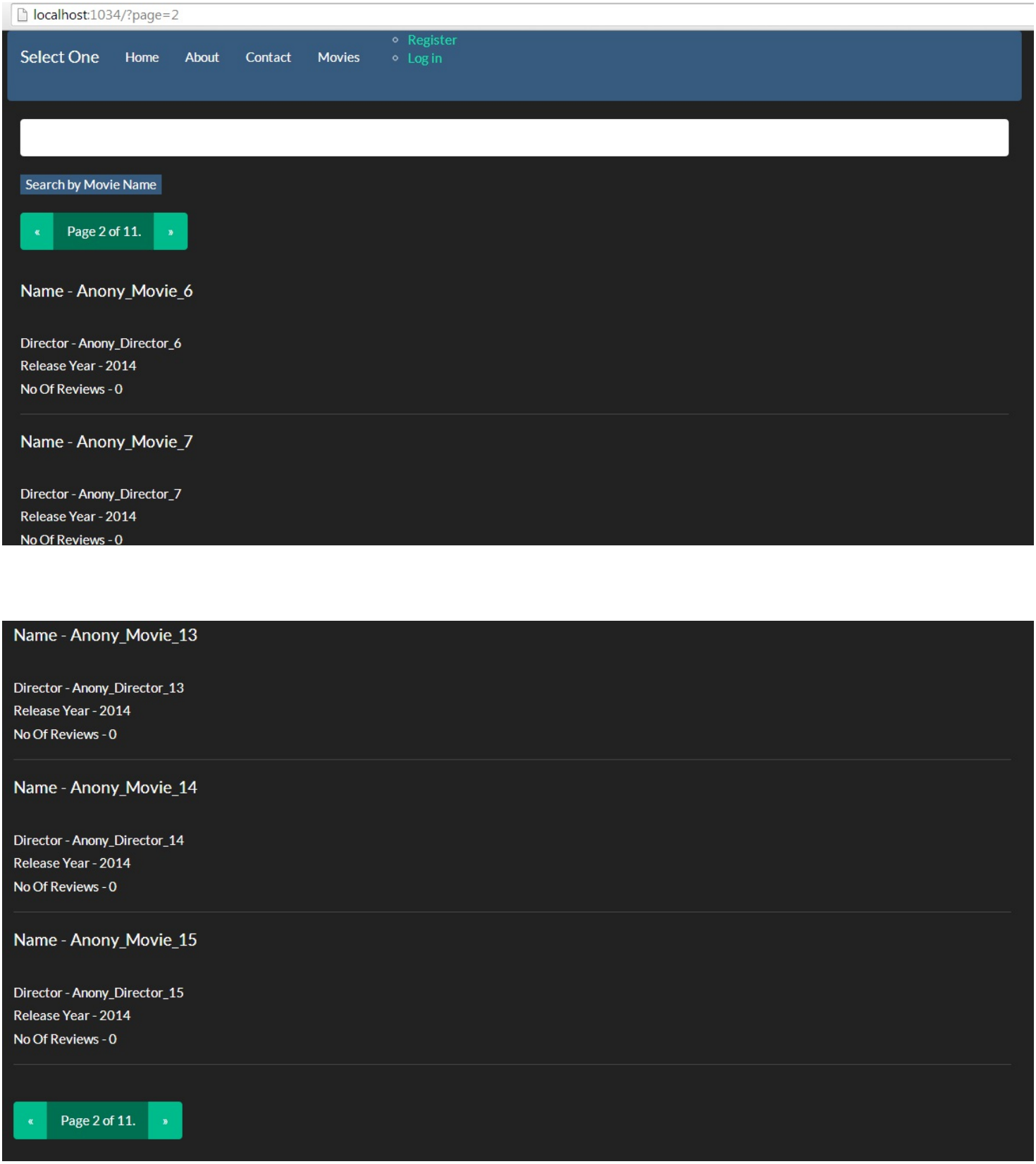
    <div class="pagedList" data-movie-target="#movieDiv">
        @Html.PagedListPager(Model, page => Url.Action("Index", new { page}),
        PagedListRenderOptions.MinimalWithPageCountText)
    </div>
         @foreach (var item in Model) {
        <div>
            <h4>Name - @item.MovieName</h4><br />
            <h5>Director - @item.DirectorName</h5>
            <h5>Release Year - @item.ReleaseYear</h5>
            <h5>No Of Reviews - @item.NoOfReviews</h5>
            <hr />
        </div>
}
    <div class="pagedList" data-movie-target="#movieDiv">
        @Html.PagedListPager(Model, page => Url.Action("Index", new { page }),
        PagedListRenderOptions.MinimalWithPageCountText)
    </div>
</div>
```

Now, as you can see in the above code I have put one section for pager. Now, obviously you could select whatever **render** options you are comfortable with. Also, I have put the paging ability both at the top of the page and at the bottom of the page. Now, when I build

the application and refresh the page, it will present me the below output.



Now, when I click on the next button, it will fetch me next 10 records from the 2nd page.



So, it's working. But, currently it's not

making the async search. So, in order to fix the same, we need to switch to the javascript file and add the async capability for paging as well. So, here what I am going to do; add "***main-content***" which is there in **Layout view** as shown below in my custom script section.

```
<div>
        <section class="main-content">
            @RenderBody()
        </section>
    </div>
```

Then, I am going to wire up the click event of this section in the **main-content** as this contains all the view as it rendered. Now, below is the script code

in its finished form. I have also put the comments wherever necessary to understand the script.

```
$(function () {

    var MovieFormSubmit = function () {
        //Grab the refernce of the form
        var $form = $(this);

        //Build the options object
        var options = {
            url: $form.attr("action"),
            type: $form.attr("method"),
            data: $form.serialize()
```

```
        };

        $.ajax(options).done(function (data) {
            var target = $($form.attr("data-movie-target"));
            target.replaceWith(data);
        });

        //To prevent the browser from doing it's defualt action means navigating
away and redrawing the complete page
        return false;
    };

    var fetchPage = function () {
        //Get the anchor tag that user clicked on
        var $anchor = $(this);

        //Extract values like Href attributes which is in the anchor tag
        var options = {
            url: $anchor.attr("href"),
     data: $("form").serialize(),
            type: "get"
        };

        //make the ajax request with options object, when the data retrieved
successfully,
        //go and find out the target and then replace the taret with the fetched
one
        $.ajax(options).done(function (data) {
            var    target    =    $anchor.parents("div.pagedList").attr("data-movie-
target");
            $(target).replaceWith(data);
        });
        return false;
    };
    //Look for Form with the name "data-movie-ajax", then wire up the submit event.
    $("form[data-movie-ajax='true']").submit(MovieFormSubmit);
    //Find the main-content and wire up the click event then filter these events
based on ".pagedList a"
    //then call the method fetchPage
    $(".main-content").on("click", ".pagedList a", fetchPage);
});
```

Now, with the above piece of changes in place, I could go ahead and test my app. So, just build and refresh the page and see if it works. So, now I can see that paging is happening asynchronously without breaking anything. With this topic, we have come to an end of this chapter. In the next chapter, we'll discuss more on security features in MVC.

Summary:-

So, in this chapter we have enabled Ajax features in our application. We have seen different Ajax Helpers in action. We have also seen how to work with Jquery to do the DOM manipulation. We

have enabled custom search capability to search the results. In the end we have added paging facility as well to enable the users to page through the list.

Chapter 7: Security with ASP.Net MVC

WHAT DO you find in this CHAPTER?

⇒ Introduction

⇒ Authentication

⇒ Membership and Forms Authentication

⇒ Authorize

⇒ Seeding Membership

⇒ Cross Site Vulnerability

⇒ OpenID and OAuth

⇒ Summary

Introduction:-

Hi Friends, in this section we'll go one step ahead by implementing security

features in our application. In this chapter we'll be working with users, roles and also see that how to use **OpenID** and **OAuth** in our application so that our users can sign in to our application by using third party authentication scheme. We'll also come across some of the most common attacks. We'll see **Cross Site Request Forgeries** and how to prevent the same. We begin the chapter by looking at different styles of authentication available in our application.

Authentication:-

Authentication of user means verifying

the identity of the user. This is really important. You might need to present your application only to the authenticated users for obvious reasons. We have couple of authentication in place like

Forms Authentication:-
Windows Authentication:-
Open ID/ OAuth:-

So, let me go ahead and discuss very 1st **Forms Based Authentication**. With forms based authentication, websites are basically responsible for presenting a page with user id and password options. With this page, user has to fill in his credentials to log in to the website.

Forms authentication basically relies on cookies. So, once the user signed in to an application, the runtime will store a cookie in the browser itself, hence they don't need to sign in again during the same browsing session. Many a time **SSL(*Secure Socket Layer*)** is introduced to make the Forms authentication secure.

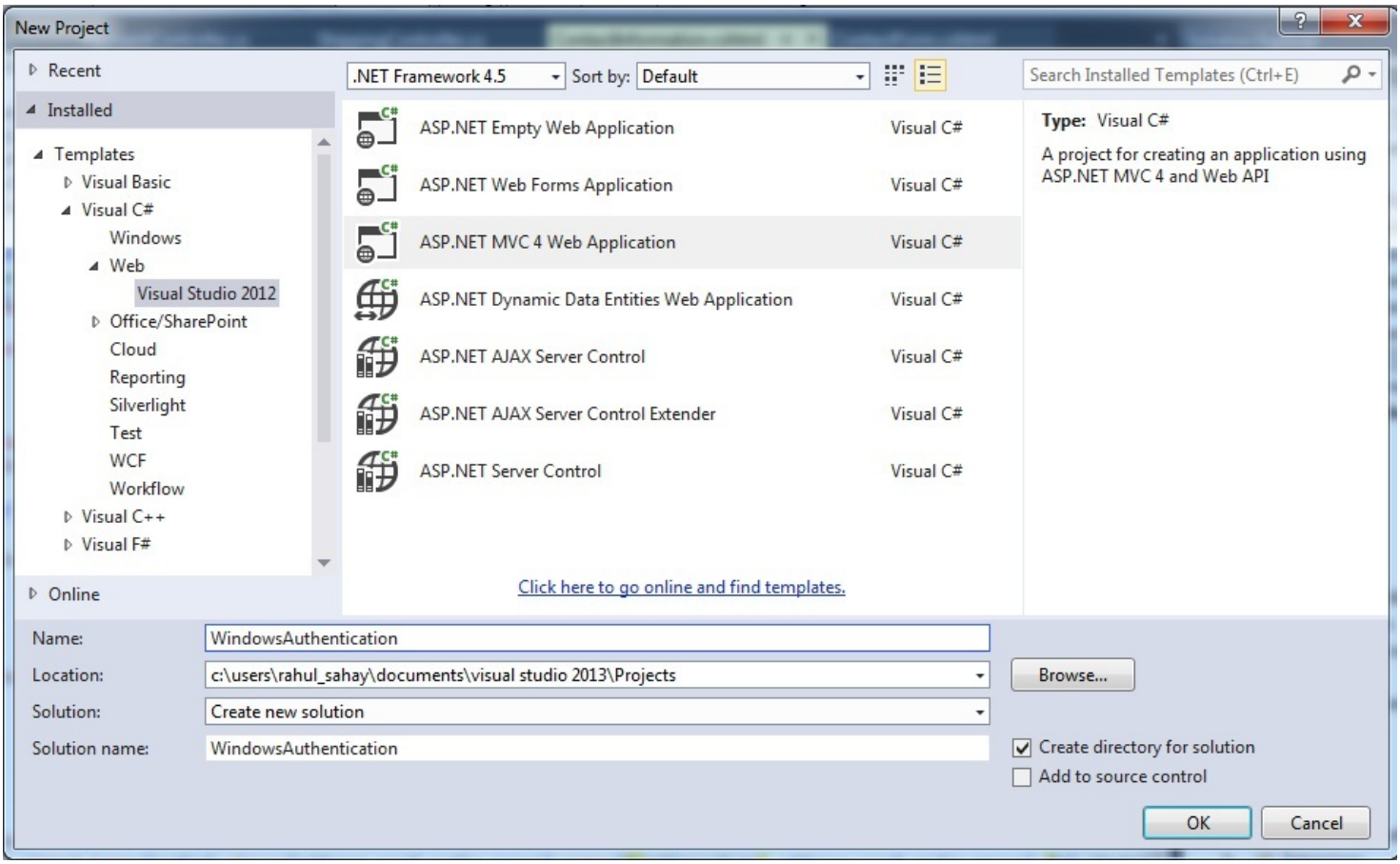
2nd type of authentication is windows authentication. It's basically used for **Intranet** applications. It's pretty much bound to the components which are provided by the Windows operating system. So, here basically all the users are get authenticated by windows active directory logged in windows domain.

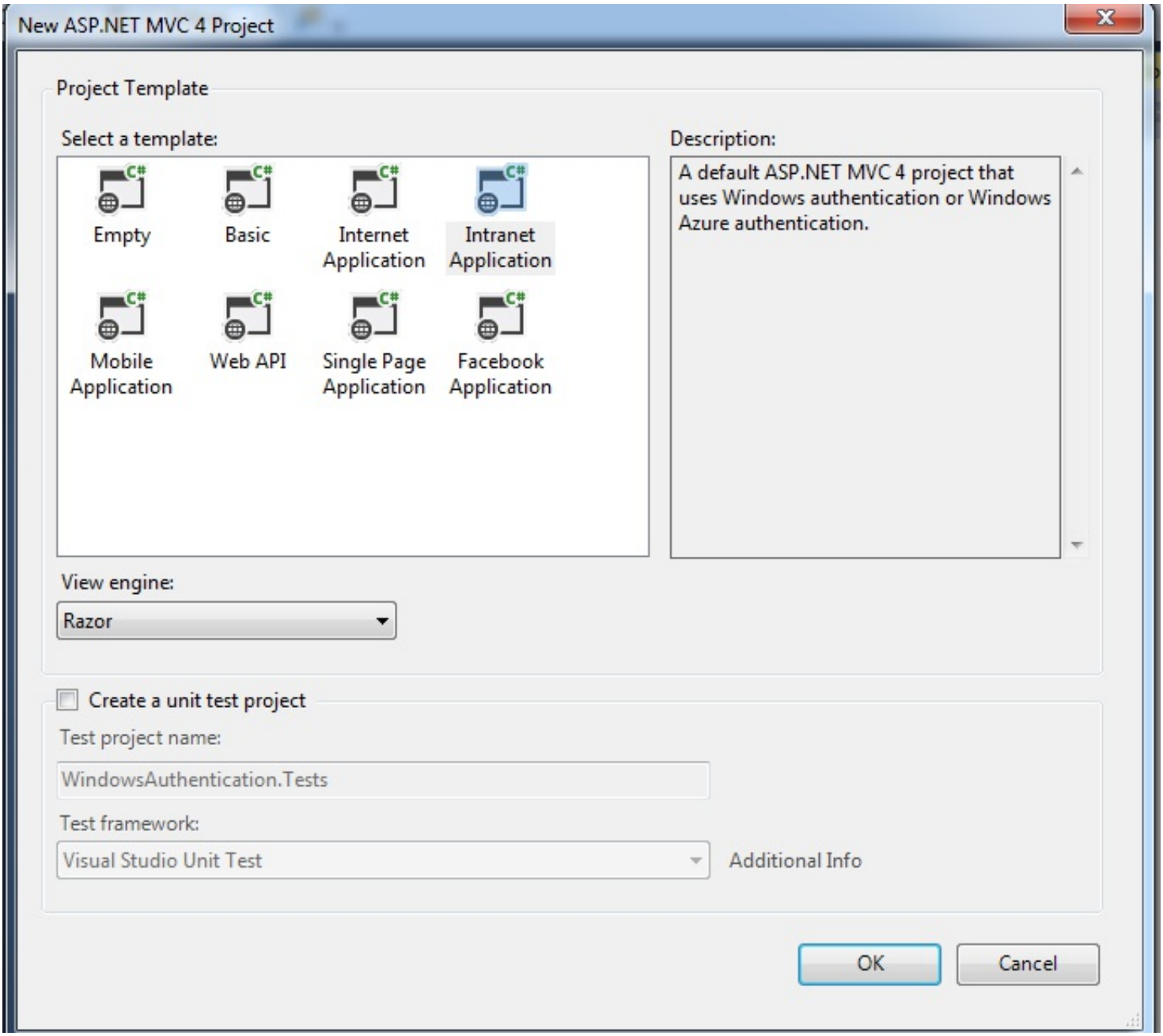
Here, users will be having the single sign on experience. As user logged in windows, user can use any facilities provided by the windows.

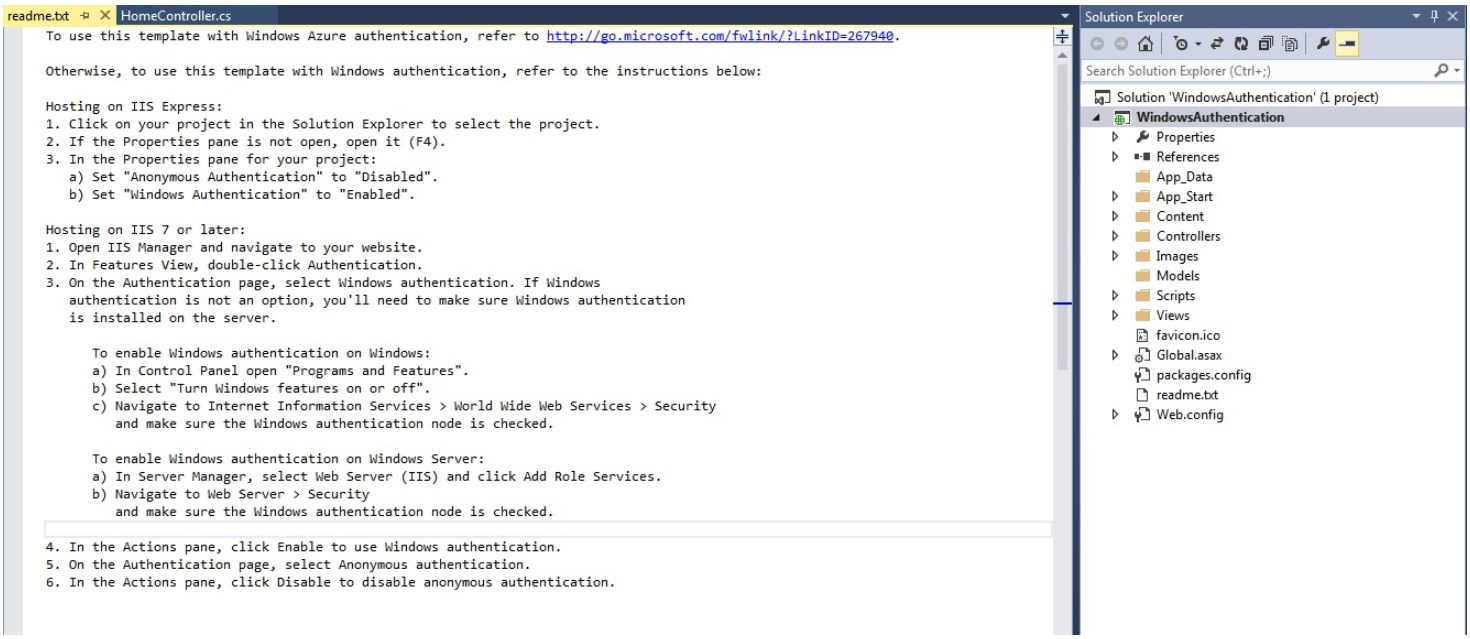
Now, the third way to authenticate users using **OpenID** or **OAuth**. These are open standards for authentication and authorization respectively. It means users don't need to create passwords with our site, rather they can get authenticated with any third party site like **Microsoft, Facebook, Google** etc and upon successful authentication, they will get redirected to our site.

Now, the application "***MovieReviews***" which we have been working on; relies

on Forms Authentication because we have selected the Internet application template. Now, in order to demonstrate that how window authentication works, let's create a new project with the name "***WindowsAuthentication***" with intranet settings as shown below.

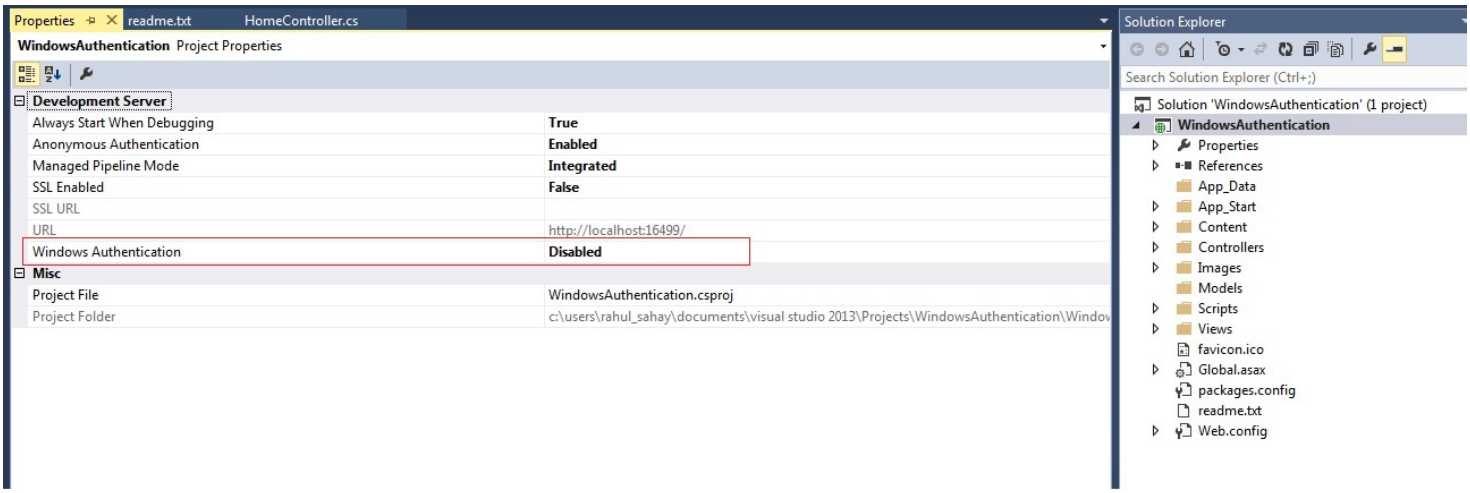



New ASP.NET MVC 4 Project
Project Template
Select a template:
Empty
Basic
Internet Application
Intranet Application
Mobile Application
Web API
Single Page Application
Facebook Application
Description:
A default ASP.NET MVC 4 project that uses Windows authentication or Windows Azure authentication.
View engine:
Razor
Create a unit test project
Test project name:
WindowsAuthentication.Tests
Test framework:
Visual Studio Unit Test
Additional Info
OK
Cancel

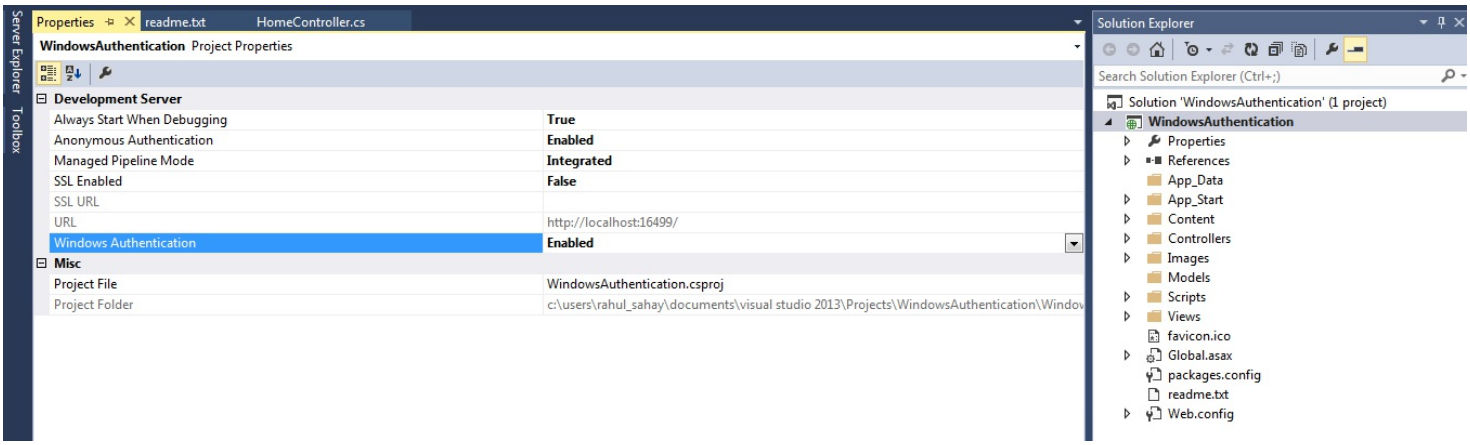


So, now the project is setup successfully and Visual studio presented me the readme.txt file. So, whenever we create any intranet application, by default it will present me this manual. It has couple of steps that one has to follow for this to work. Now, in order to setup windows authentication, I need to set the properties, it's not the project property from where I can set this up. Either I can press F4 key by highlighting the project

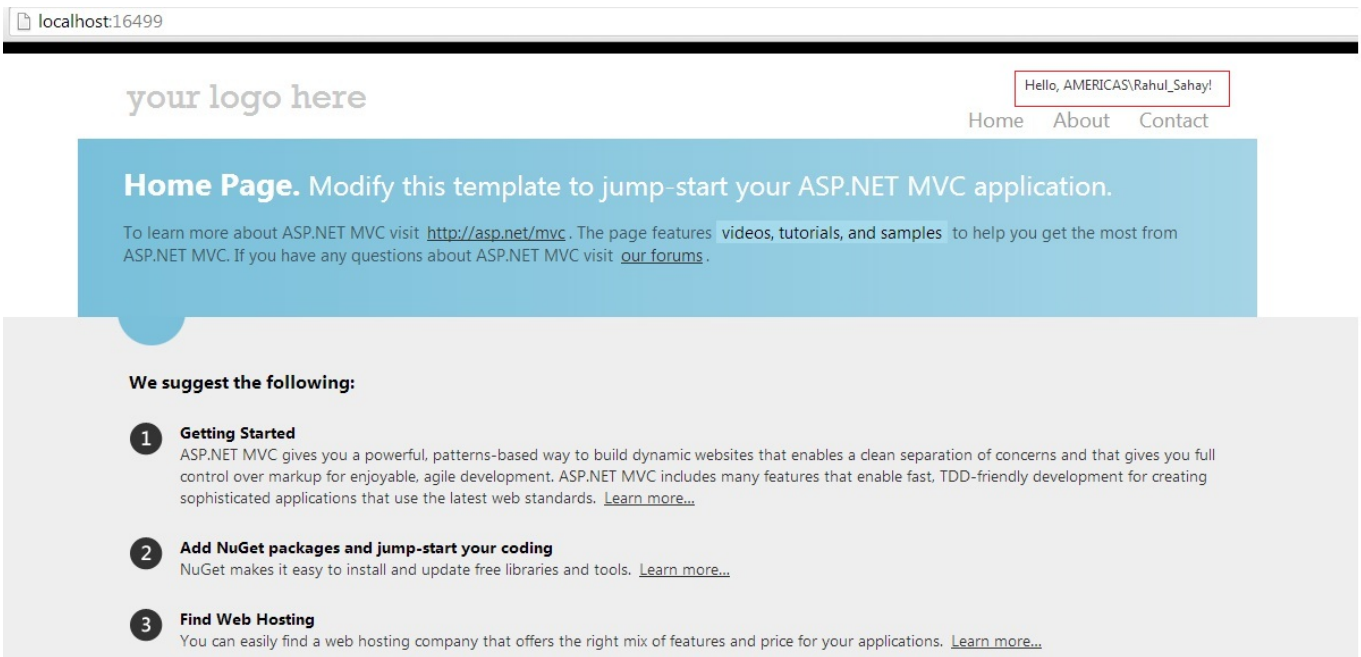
in the solution explorer in visual studio to go in the properties window or in the (**View → Properties Window**). Upon clicking it will open me the below window.



Now, as you can see in the above screen shot that Windows Authentication is disabled. Now, to set up windows authentication, I just need to enable this flag as shown below.



Now, with this property setup, I am good to go. So, now when I run the application I should be getting my name with my domain name flagged in there.



Now, if I inspect the code for the same

in Layout views, it checks the "***@User.Identity.Name***" to pick-up the user from the **Domain Directory**. However, the same property will be used in Forms Authentication as well. With the above piece of code, we could verify whether the users are authenticated or not and many more related with user and authentication. Other difference you might notice in the web.config file where in authentication section is marked as windows.

```
<authentication mode="Windows" />
```

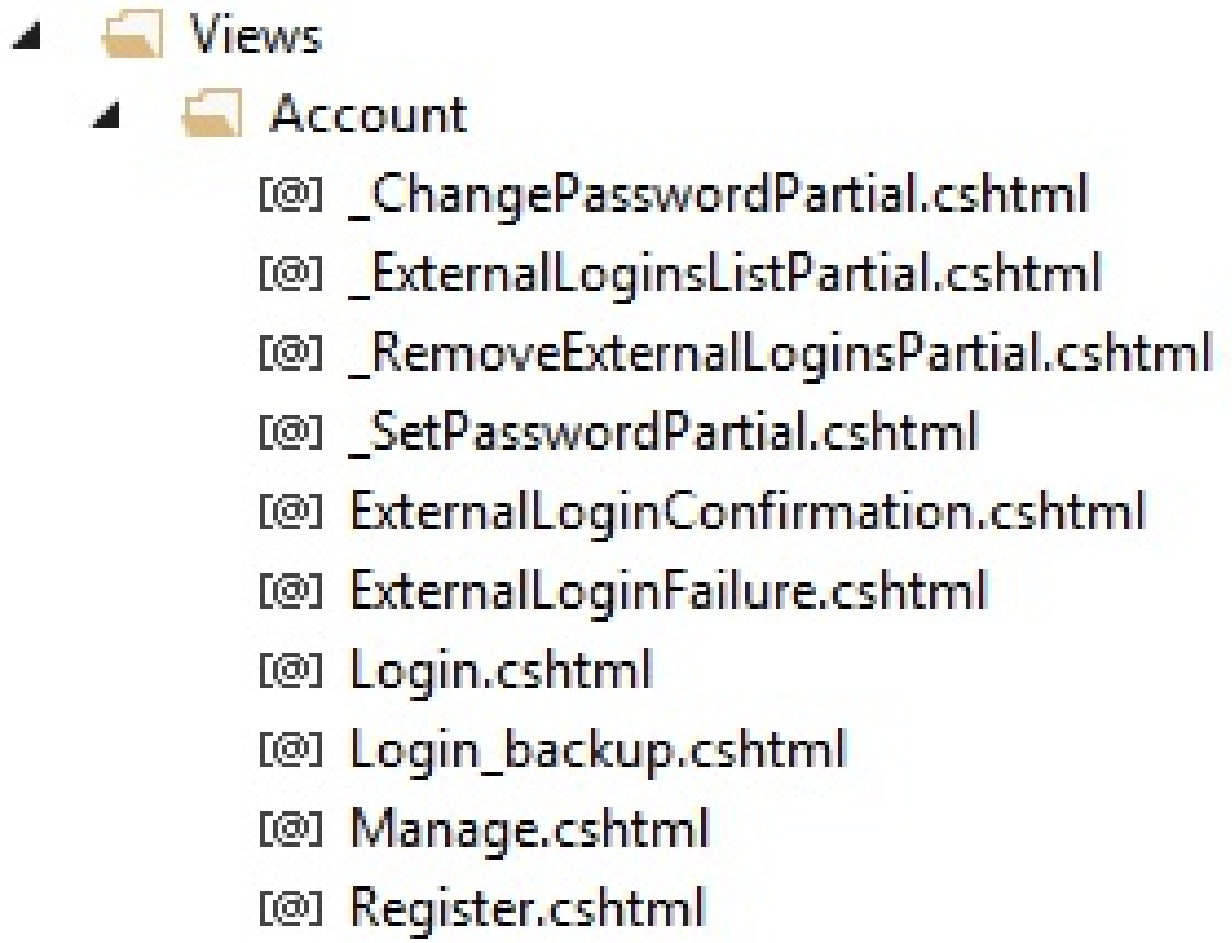
In case if it would have been Forms based authentication, then in that case it would have been "***Forms***".

Now, the next thing is Forms based authentication. So, basically what happens in Forms Authentication is like let's suppose on your site, there is some restricted page which needs user authentication, so when you try to invoke that page from the **URL**, application will redirect you to the login page where in you will provide your credentials and upon authentication, it redirects you to required page. Again, ASP.Net architecture works here in such a way, that it saves the request in the "***returnUrl***" parameter in **URL** itself and then when user get authenticated, user will get redirected to that particular page. ASP.Net also gives browser a cookie for the application.

So, a cookie is nothing but a piece of data which the browser will hold. And once the browser holds the cookie for the site, it will continue to send the cookie along with every request which came to the site as long as browsing session is not closed. Now, inside this cookie, there is some cryptic information letting **ASP.NET** know that user is already authenticated and hence user can access those restricted pages which required authentication. So, this is how Forms Authentication works in general.

Now, let me go ahead and explain the setting which we have been using so far in our application. Since we have created our project based on internet

application, so visual studio also created default structure needed for the user to **Sign Up**, then **Sign In**, **Change Password** and many more basic stuffs. So, all these basic stuffs are provided by the "***AccountController***" in our application; which will obviously base on separate user actions will render different views as shown below in the screen shot.



And also associated to this controller there is one model associated which you could find in the Models folder with the name “AccountModels”. Now, if you see inside the Account controller code as shown below, you will come to know it's using a class with the name “WebSecurity”

```
[HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public ActionResult Login(LoginModel model, string returnUrl)
        {


            if(ModelState.IsValid && WebSecurity.Login(model.UserName,
model.Password, persistCookie: model.RememberMe))
            {
                return RedirectToLocal(returnUrl);
            }

            // If we got this far, something failed, redisplay form
            ModelState.AddModelError("", "The user name or password provided is
incorrect.");
            return View(model);
        }
```

Now, **WebSecurity** is basically inherited from Microsoft Library named "***WebMatrix***". So, **WebMatrix** is another tool from **Microsoft** to build websites. Here ASP.Net MVC just used this technology to implement Forms Authentication. So, now it's WebSecurity class which will talk to account controller and validate the user.

Now, in return **WebSecurity** talks to one component called "***SimpleMembershipProvider***". So, ultimately, Membership Provider doing all the hard work of storing user's information in the database. Now, one point to note here, this is just the default settings which are given to developers by Visual Studio. Microsoft never mandates like it has to be like this only. So, obviously you can go ahead and customize the controller and views as you like. But, again it gives a good starting point to go ahead with your application. I'll show some of the steps that how to customize later in this chapter.

Membership and Forms Authentication:-

Now, before we dig inside the Account controller's code and see how it works, let me explain few more steps that what's happening behind the scenes especially at the database layer. So, here I'll be customizing the information which is being stored at the user level in the database. So, the very step in this case is coming up and looking for the file "***InitializeSimpleMembershipAttribute*** in the "***Filters***" folder. So, if we open this file, there's a bunch of complex code is there, basically generic code. It is written in such a fashion, that either we

could use Forms Authentication or some other Authentication. Hence, Forms Authentication in here is initialized in a lazy manner. But, since we have already decided that we need to use the Forms Authentication, hence there is no need to use the complete bunch of code. The line of code which really makes sense over here is pasted below.

```
WebSecurity.InitializeDatabaseConnection("DefaultConnection","UserProfile",
"UserId", "UserName", autoCreateTables: true);
```

Here, in the above case WebSecurity is initializing the database with some of the default parameters as shown above. The 1st parameter which it takes is the connection string, now this is the same connection string which we are using

here for our application as well. 2nd parameter which it takes is the name of the table which will contain the user information.3rd parameter is the column in the table which contains the primary key value and the 4th parameter is the column which contains the user name value.

Now, obviously you can go ahead and customize this table as well; you can have as many columns as you want to store user profile related values in the table. So, if we are 100% sure that we are using Forms Authentication, then we take this piece code from here and use the in "***global.asax***" file. So, this piece of code we don't need any more in this

file. So, now my global.asax file will look like below. Now, if you want to be consistent the way global.asax file code written, then in that case you can wrap up this piece of code in some class and call the Register method from here. But, for now it's ok to have the same here itself.

```
public class MvcApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            WebSecurity.InitializeDatabaseConnection("DefaultConnection",
"UserProfile", "UserId", "UserName", autoCreateTables: true);
            AreaRegistration.RegisterAllAreas();

            WebApiConfig.Register(GlobalConfiguration.Configuration);
            FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            BundleConfig.RegisterBundles(BundleTable.Bundles);
            AuthConfig.RegisterAuth();
        }
    }
```

Now, since we have moved this piece of

code in our global.asax file. So, we don't need "***InitializeSimpleMembershipAttribute*** file any more, hence we can go ahead and delete the same. Now, there is one more thing which I need to do, inside the models folder there is file "***AccountModels.cs***" as shown below.

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Data.Entity;
using System.Globalization;
using System.Web.Security;

namespace MovieReview.Models
{
    public class UsersContext : DbContext
    {
        public UsersContext()
```

```
            : base("DefaultConnection")
        {
        }

        public DbSet<UserProfile> UserProfiles { get; set; }
    }

    [Table("UserProfile")]
    public class UserProfile
    {
        [Key]
        [DatabaseGeneratedAttribute(DatabaseGeneratedOption.Identity)]
        public int UserId { get; set; }
        public string UserName { get; set; }
    }

    public class RegisterExternalLoginModel
    {
        [Required]
        [Display(Name = "User name")]
        public string UserName { get; set; }

        public string ExternalLoginData { get; set; }
    }

    public class LocalPasswordModel
    {
        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Current password")]
        public string OldPassword { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters
long.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "New password")]
        public string NewPassword { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Confirm new password")]
        [Compare("NewPassword", ErrorMessage = "The new password and confirmation
password do not match.")]
        public string ConfirmPassword { get; set; }
    }
```

```
    public class LoginModel
    {
        [Required]
        [Display(Name = "User name")]
        public string UserName { get; set; }

        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]
        public string Password { get; set; }

        [Display(Name = "Remember me?")]
        public bool RememberMe { get; set; }
    }

    public class RegisterModel
    {
        [Required]
        [Display(Name = "User name")]
        public string UserName { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters
long.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]
        public string Password { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Confirm password")]
        [Compare("Password",  ErrorMessage  =  "The  password  and  confirmation
password do not match.")]
        public string ConfirmPassword { get; set; }
    }

    public class ExternalLogin
    {
        public string Provider { get; set; }
        public string ProviderDisplayName { get; set; }
        public string ProviderUserId { get; set; }
    }
}
```

Now, if you see the snippet closely you will recognize that there is a "***UserContext***" class which is derived from "***DbContext***" .But, now we want to control this, now we want to have this by our "***db context***" class which we have already created. So, I am going to delete this section. Now, just below to that section, you could see there is an auto generated code for the "***UserProfile***" table, which we want to customize. So, in order to do the same, I just cut that piece of code and put the same in a new class file named as "***UserProfile***" under the models folder and paste the code in there. Now, in this class I need to bring in the name space for the "***Table***" annotation. So, below is

the snippet of the user profile table in its finished form.

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Web;

namespace MovieReview.Models
{
    [Table("UserProfile")]
    public class UserProfile
    {
        [Key]
        [DatabaseGeneratedAttribute(DatabaseGeneratedOption.Identity)]
        public int UserId { get; set; }
        public string UserName { get; set; }

        public string FavoriteColor { get; set; }
    }
}
```

So, if you see the above snippet I have added one custom property saying favorite color for the user in the User profile table. Now, there is "***[Key]***"

annotation there before the "***UserId***" column which is telling the Entity Framework that this is primary key in the table. Also, associated to that there is identity field which means that database will manage the new values coming in the database by just auto incrementing the same by 1. Now, I need to define the way to get into my own "***DbContext***" class. So, I'll go into the file "***MovieDb.cs***" and add one property for the same.

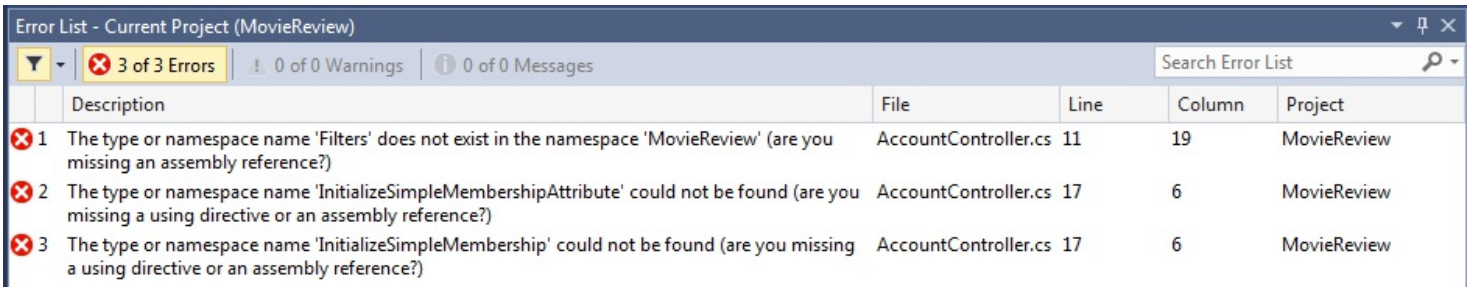
```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Security.AccessControl;
using System.Web;
```

```
namespace MovieReview.Models
{
    public class MovieDb :DbContext
    {
        public MovieDb()
            : base("name=DefaultConnection")
        {

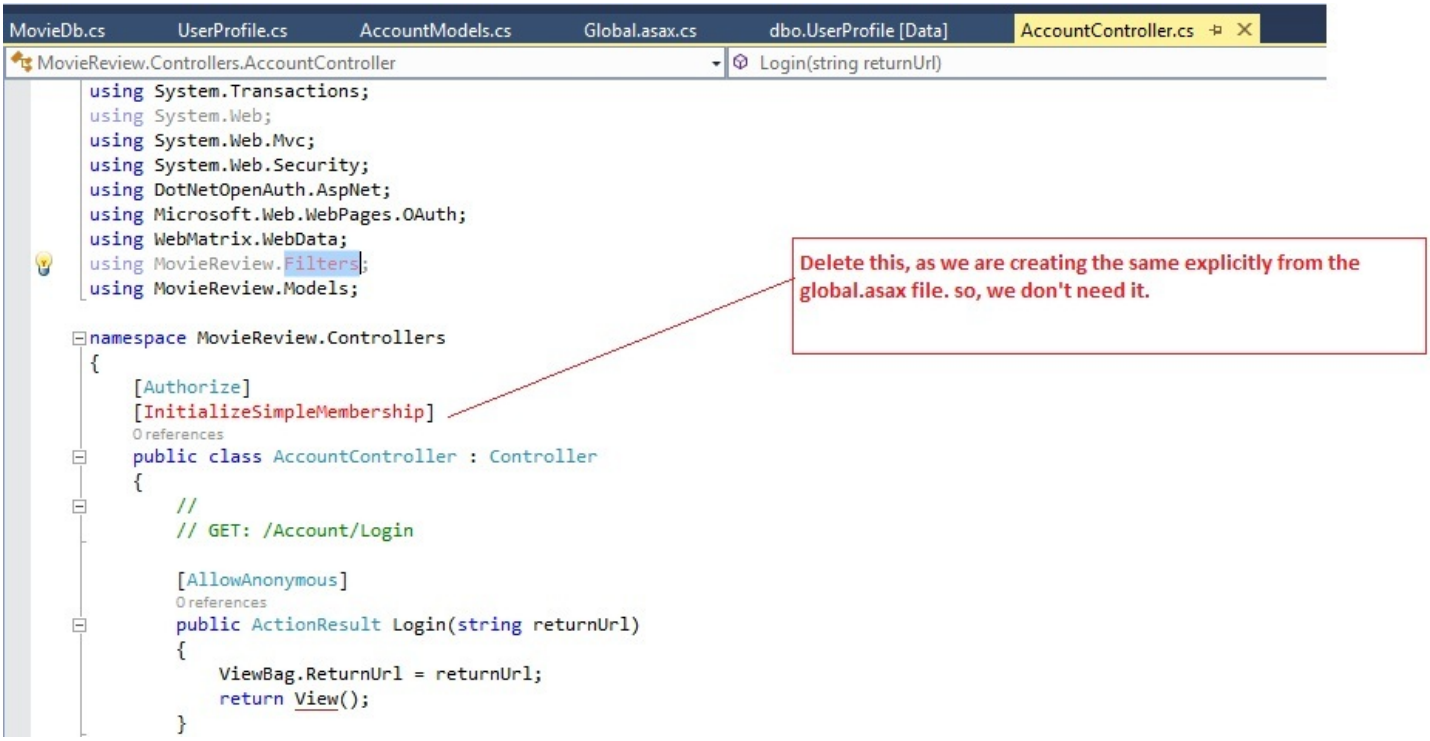
        }

        public DbSet<UserProfile> UserProfiles { get; set; }
        public DbSet<Movie> Movies { get; set; }
        public DbSet<MovieReviews> MovieReviews { get; set; }
    }
}
```

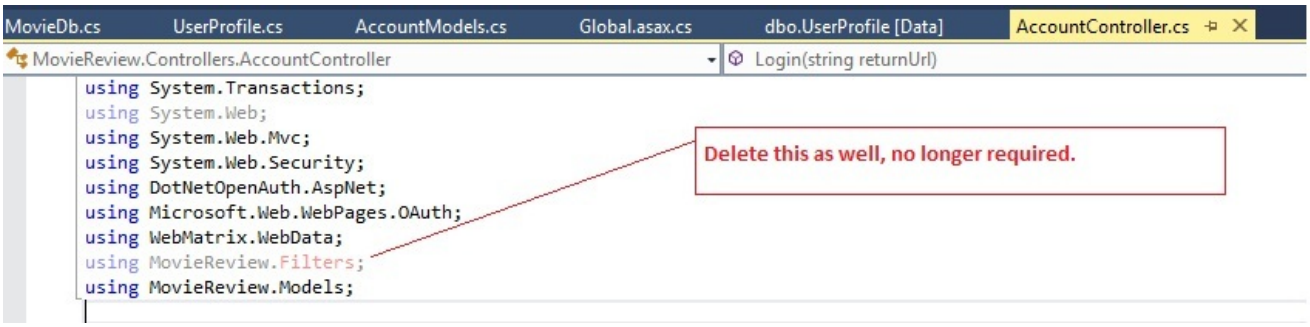
Now, since we have made a change in our context file, hence we need to do the schema migration in order to get the effect. But, before that we need to fix few errors. So, when we do build it will list me few errors as shown below because we have deleted the file from the filters table.



Now, these errors are quite educational, so when you double click on the same, you will get the below thing

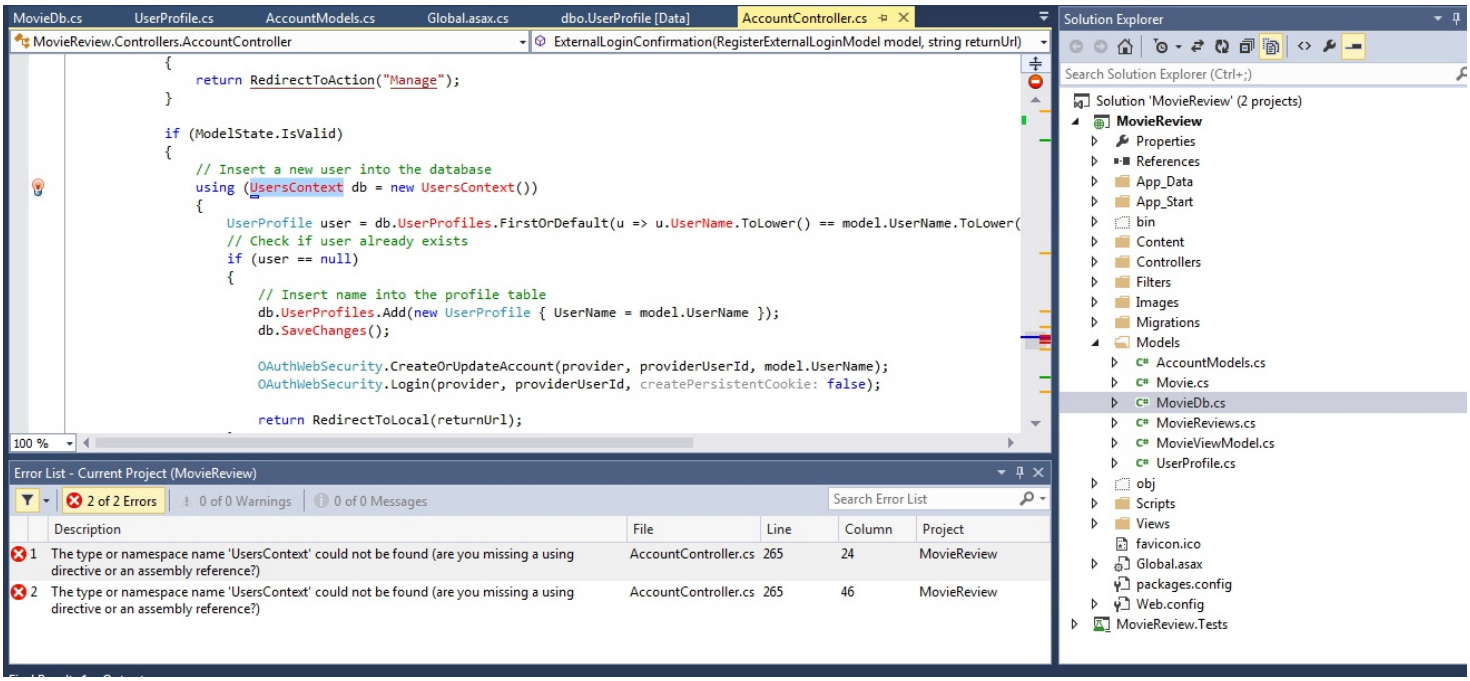


Now, when you click on the 2nd error you will reach there



Now for the below error, you just need to modify the context name, here you need to specify our context name which

we have been using in our application.

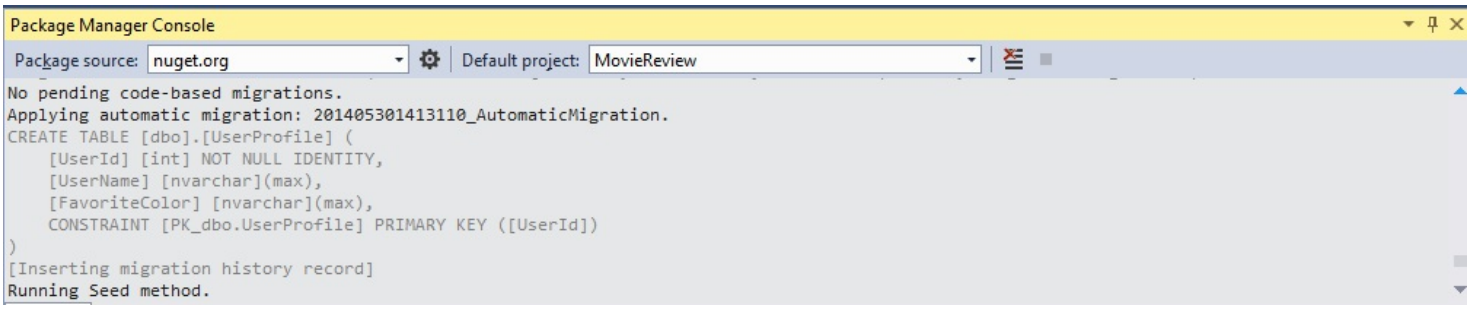


Now, the modified code for the above error in its finished form is pasted below.

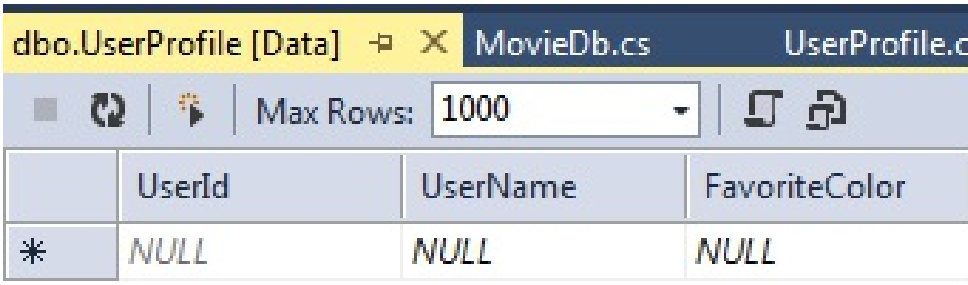
```
if (ModelState.IsValid)
            {
                // Insert a new user into the database
                using (var db = new MovieDb())
                {
                    UserProfile user=  db.UserProfiles.FirstOrDefau
u.UserName.ToLower() == model.UserName.ToLower());
                    // Check if user already exists
                    if (user == null)
                    {
                        // Insert name into the profile table
                        db.UserProfiles.Add(new UserProfile   {User
model.UserName });
                        db.SaveChanges();

                        OAuthWebSecurity.CreateOrUpdateAccount(prov
providerUserId, model.UserName);
```

So, up there in that section I have used my context name "***MovieDb***". Now when I build the same it will succeed. Here, I can go ahead and open the package manage console to migrate the modified schema with the Powershell Command "Update-Database -Verbose". Now, upon successful completion, it will present me the below output

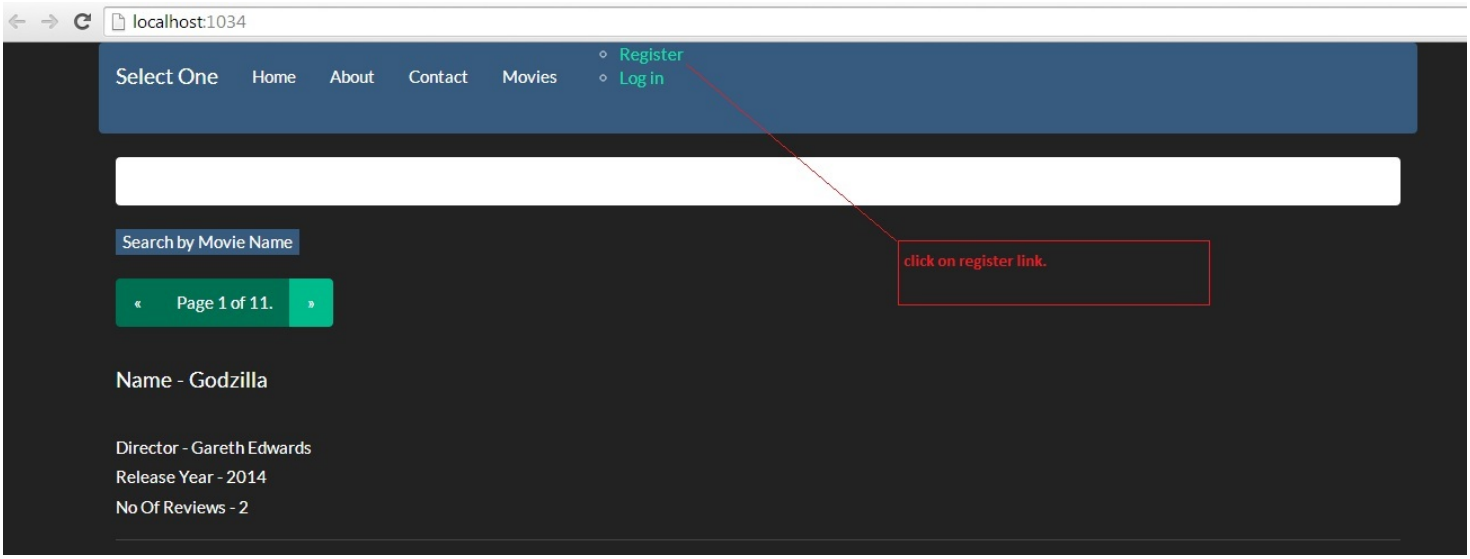


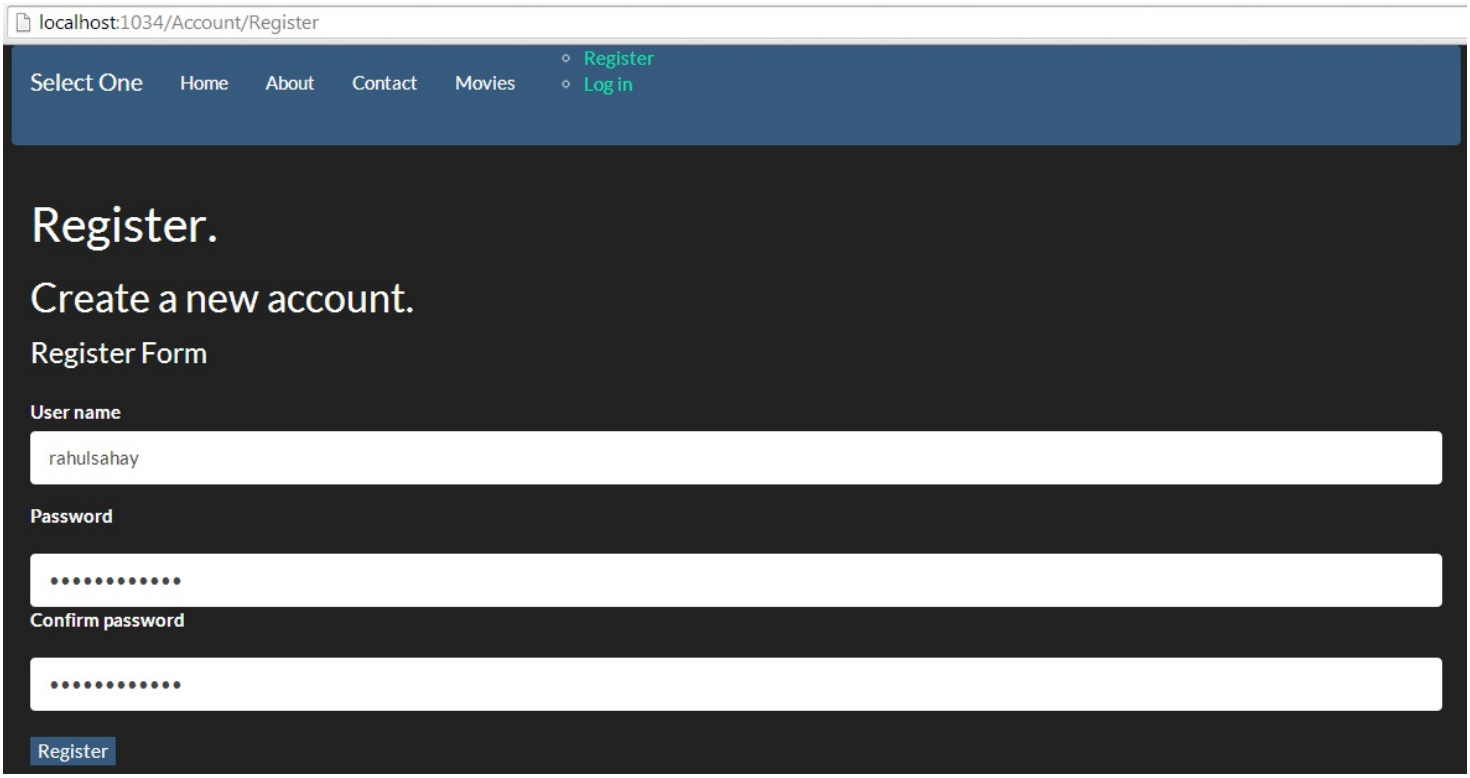
So, in the above screen shot you can see that, it has added new User profile table with my custom field in there. Now, when I go ahead and refresh the database schema, I should be able to see the newly added field in the table.



Now, let's go ahead and look at the piece that how Forms Authentication works in our application. What I will do, I'll go

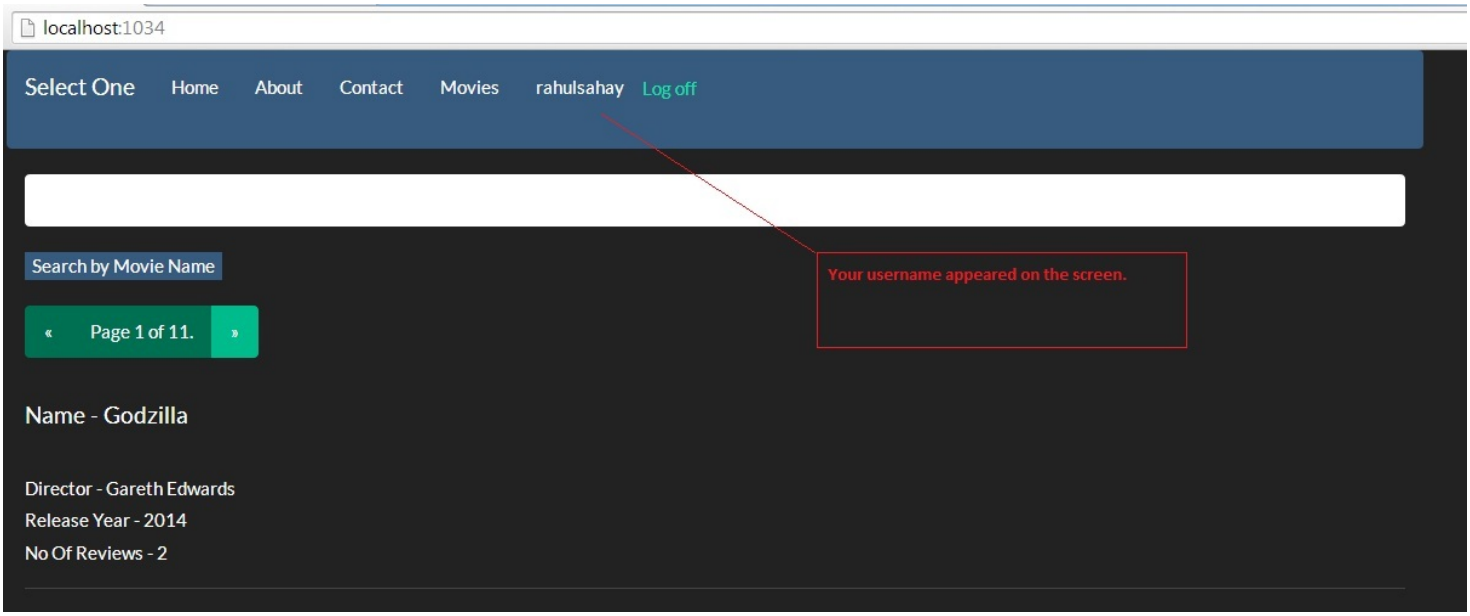
ahead and click on the register link shown below and register myself.



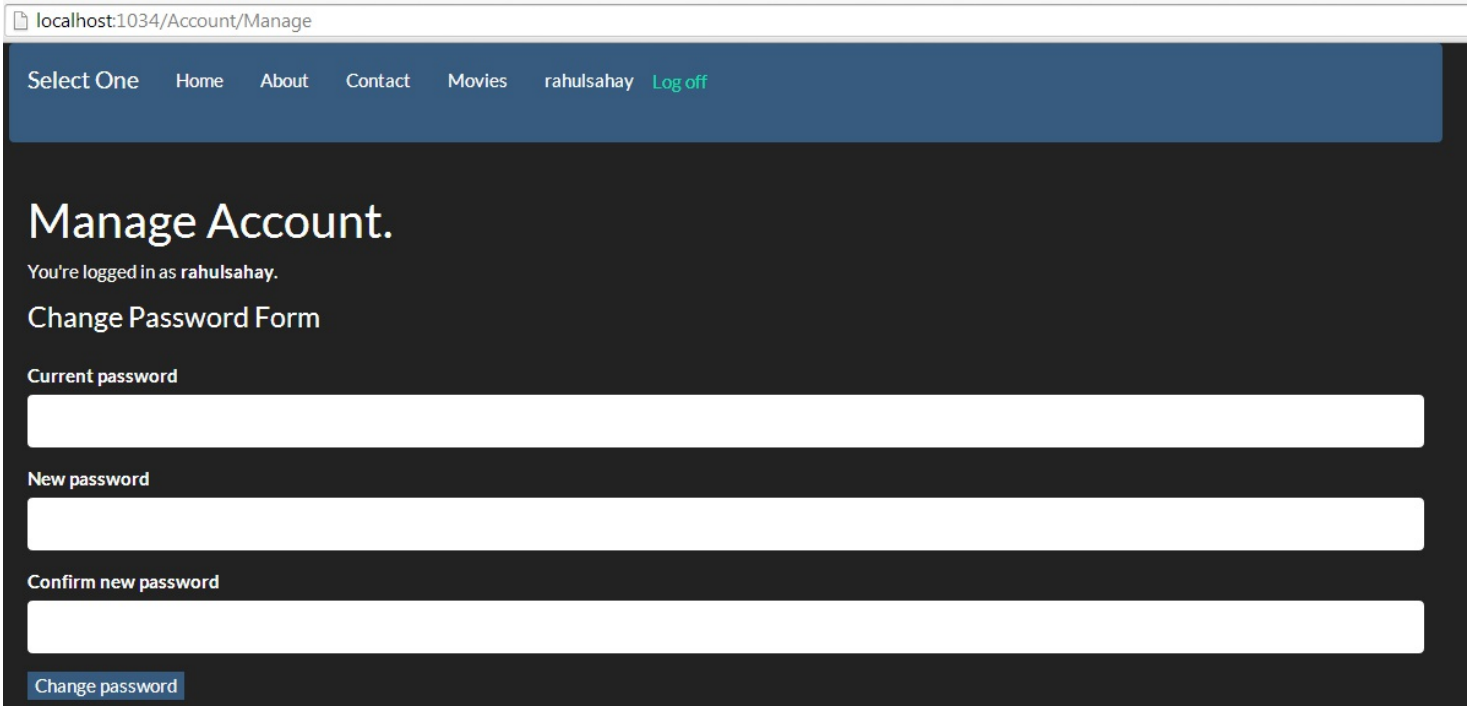


And then my name will appear on the

page as shown below and that's done using "***User.Identity.Name***" property.



Now, at this point I can manage my account to change my password by just clicking on my name as shown below:-



I can also go ahead and log off myself by clicking on the link shown in the page. So, all the basic stuffs I would be able to do with the site. All of these functionalities are provided to us by the Internet application template which we have selected for creating the application. Now, the **Register Form** from where I have registered myself in the application sits in the **Account**

folder under **Views** folder. Now, below is the code for same. However, I have modified the code a bit to fix the look and feel of the website.

```
@model MovieReview.Models.RegisterModel
@{
    ViewBag.Title = "Register";
}

<hgroup class="title">
    <h1>@ViewBag.Title.</h1>
    <h2>Create a new account.</h2>
</hgroup>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()
    @Html.ValidationSummary()

    <form class="form-horizontal">

        <fieldset>
            <legend>Register Form</legend>

            <div class="form-group">
```

```
                @Html.LabelFor(m => m.UserName)
                <div class="form-control">
                    @Html.TextBoxFor(m => m.UserName)
                </div>
            </div>
            <div class="form-group">
                @Html.LabelFor(m => m.Password)
            </div>
            <div class="form-control">
                @Html.PasswordFor(m => m.Password)
            </div>

            <div class="form-group">
                @Html.LabelFor(m => m.ConfirmPassword)
            </div>
            <div class="form-control">
                @Html.PasswordFor(m => m.ConfirmPassword)
            </div>
            <p>
                <br />
                <input type="submit" class="btn-primary" value="Register" />
            </p>
        </fieldset>
    </form>
}

@section Scripts {

}
```

Now, this is the place where in you can force user to provide their favorite color while registering themselves with the application. **Favorite Color** is the

custom property which we have added in **User Profile** table. Now, this form is getting rendered from the Account controller's Register action as shown below. So, by just looking at the piece of code, you should be now able to identify that what this piece of code is doing here.

```
//
        // GET: /Account/Register

        [AllowAnonymous]
        public ActionResult Register()
        {
            return View();
        }
```

And we also have Register method as

shown below with Post request which takes our input from the **Form** and commit the same in the database. Below is the piece of code which is doing the same.

```
        [ValidateAntiForgeryToken]
        public ActionResult Register(RegisterModel model)
        {
            if (ModelState.IsValid)
            {
                // Attempt to register the user
                try
                {
                    WebSecurity.CreateUserAndAccount(model.UserName,
model.Password);
                    WebSecurity.Login(model.UserName, model.Password);
                    return RedirectToAction("Index", "Home");
                }
                catch (MembershipCreateUserException e)
                {
                    ModelState.AddModelError("", ErrorCodeToString(e.StatusCode));
                }
            }

            // If we got this far, something failed, redisplay form
            return View(model);
        }
```

Now, as you can see in this method, we have "***RegisterModel***" like a View Model contains only the information about the user as shown below which is there in the "***AccountModels.cs***" class.

```
public class RegisterModel
    {
        [Required]
        [Display(Name = "User name")]
        public string UserName { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters
long.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]
```

Now, as you can see in this method, we have "***RegisterModel***" like a View

Model contains only the information about the user as shown below which is there in the "***AccountModels.cs***" class.

```
public class RegisterModel
    {
        [Required]
        [Display(Name = "User name")]
        public string UserName { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters
long.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]

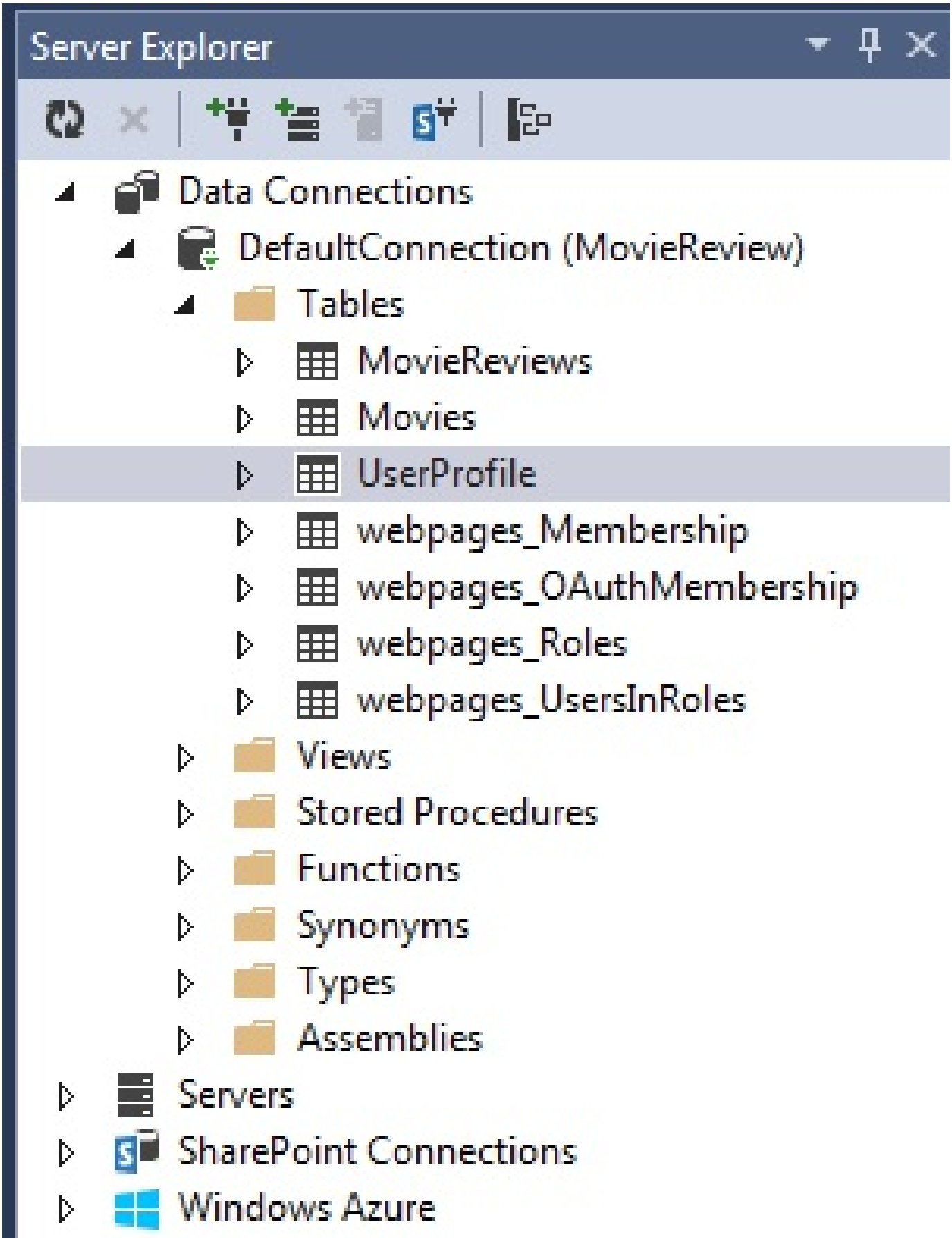

        public string Password { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Confirm password")]
        [Compare("Password",  ErrorMessage  =  "The  password  and  confirmation
password do not match.")]
        public string ConfirmPassword { get; set; }
    }
```

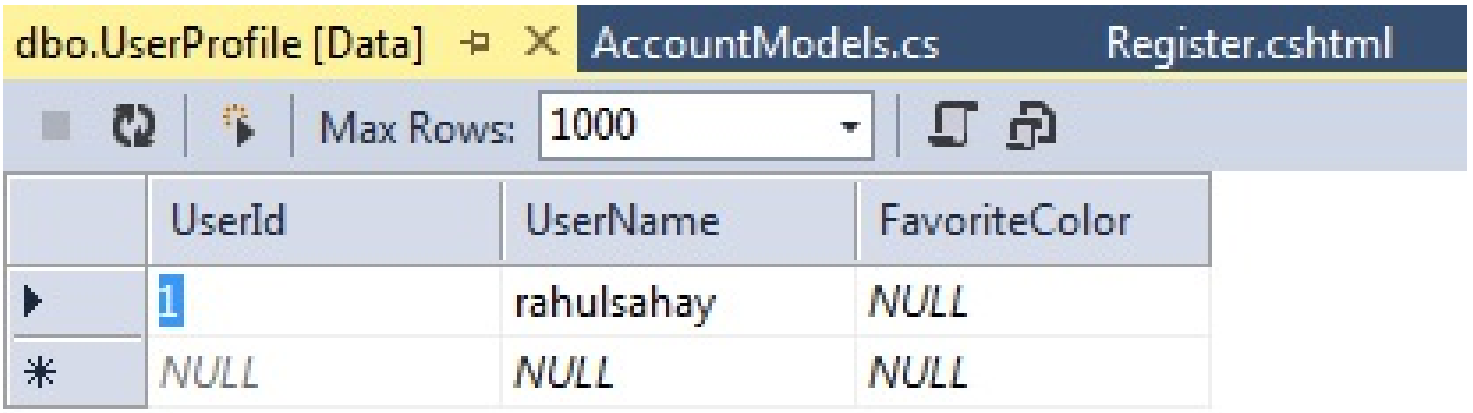
Now, in the register method, it's using the call for "***WebSecurity***".

WebSecurity is basically a wrapper around membership functionality. It takes

care of all the basic stuffs like data access, security and all the other stuffs which you need for Forms based authentication. Then, I can see that there is a call for "***CreateUserAndAccount***", This is the call which will insert my record in the database. Now, if you remember that we have modified the default context with our context and specified the same under the connection "***DefaultConnection***" which is being used by our application to manage "***Movies***" and "***MovieReviews***" table. Now, when I open the Server explorer and look our database, it will list below shown tables.

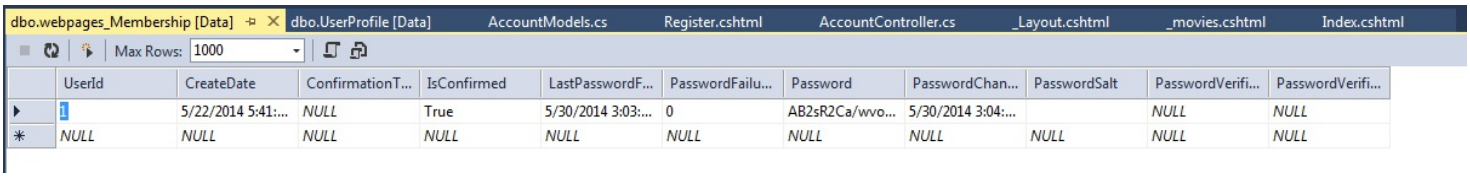

Server Explorer
Data Connections
DefaultConnection (MovieReview)
Tables
MovieReviews
Movies
UserProfile
webpages_Membership
webpages_OAuthMembership
webpages_Roles
webpages_UsersInRoles
Views
Stored Procedures
Functions
Synonyms
Types
Assemblies
Servers
SharePoint Connections
Windows Azure

So, 1st three tables are those tables which we have created and customized and hence own its behavior. However, other tables which are started by "***webpages***" are created by "***WebSecurity***" class. Now, if I open the user profile table, it will present me the below information.



It created the table exactly same how we defined. Since, we haven't provided the option for "***FavoriteColor***" while registering the user in the view, hence it

is "***NULL***" there. However, "***UserName***" is exactly the same how we entered while registering. Now, if open the table "***webpages_Membership***", I can see that there is one column "***UserId***" with the same value what we have in "***UserProfile***" table. So, now this column could be used to join these two tables for any specific query.



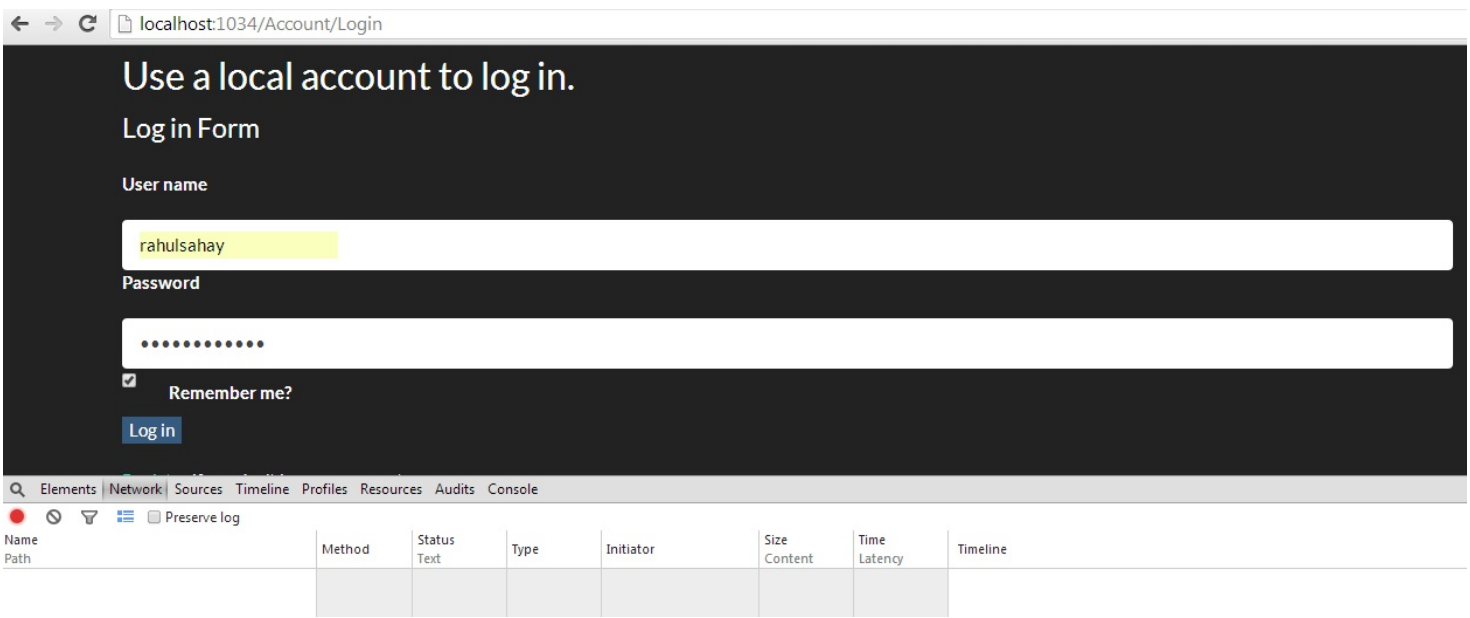
UserId	CreateDate	ConfirmationT...	IsConfirmed	LastPasswordF...	PasswordFailu...	Password	PasswordChan...	PasswordSalt	PasswordVerifi...	PasswordVerifi...
1	5/22/2014 5:41:...	NULL	True	5/30/2014 3:03:...	0	AB2sR2Ca/wvo...	5/30/2014 3:04:...		NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Now, as you can see in the above screen shot it has many other important columns which keep track of user logging activities. If you look closely, it stores the password in cryptic format.

WebSecurity stores the password in hashed format which is good for user security. Since we have seen that how and where the user data is getting stored, let's return to our Account Controller and see the next call where in its making the call to "***Login***" in the Register Post action. However, you may see the same call in Login method, when user clicks on the Login button.

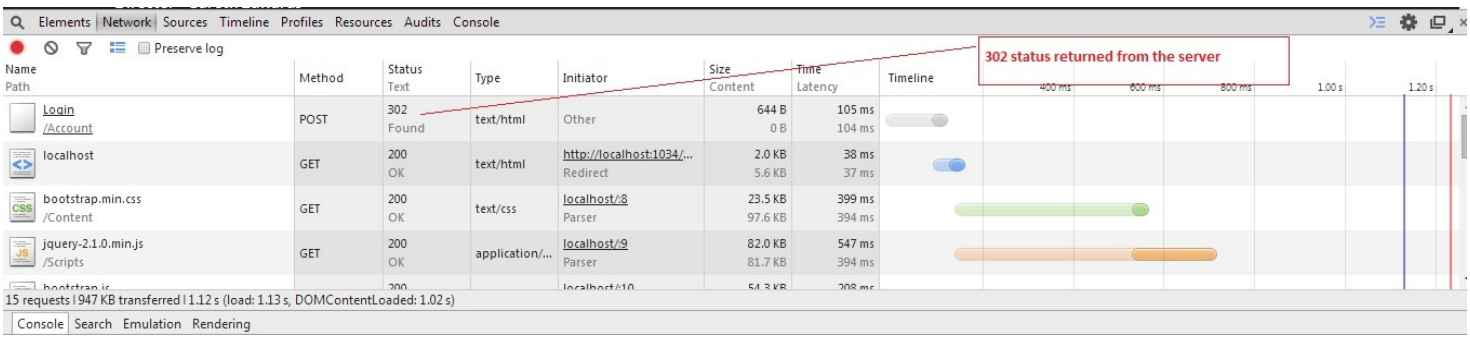
It's simply taking your username and password and passing the same to WebSecurity for validation. So, here if the **Model State** is valid and we get authenticated by the WebSecurity, then in that case we get redirected to the **Return URL**. So, here "***WebSecurity.Login***"

takes these two input and validate the same with the record stored in the database, and if they match, it will issue a cookie to the browser we discussed earlier. Now, in order to explain the same in detail, let's open chrome debugger tool and open the network tab as shown below.



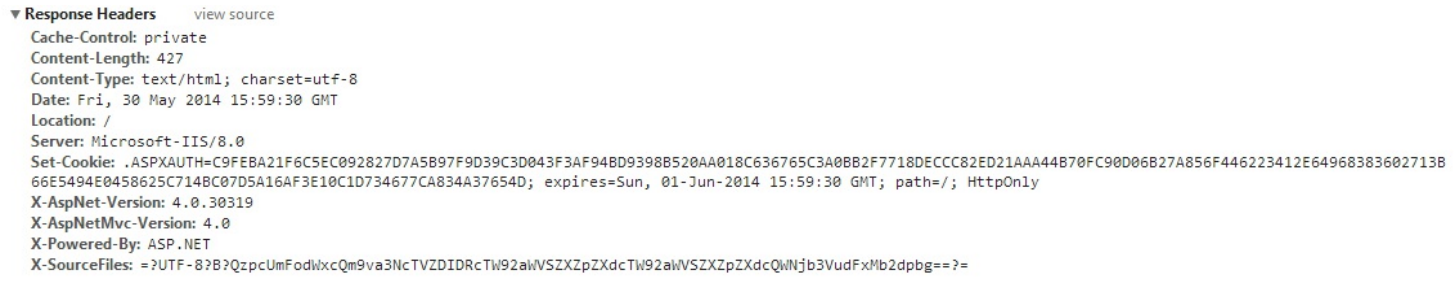
Now, as you see in the above screen shot, I have entered my credentials and

checked on the "***Remember me?***" button, so that browser can remember me for that I am authenticated for the same browsing session. Now, when I click on "***Log in***" button, it will show the below output.



So, what happened here, it determined that I logged in successfully and hence returned a 302 result which is a **HTTP Redirect Code**. Hence, it redirected me to the home page, which you can see in the 2nd result just after POST 302. Now, if open up this response from the

webserver and go to the headers section as shown below, the important thing here is "***Set-Cookie***" header which is setting the cookie with the name starting with "***.AXPAUTH***".



But, the other thing which I would like to show here, above this section the "***FormData***" section, which is also showing my **Username** and **Password**.



So, as you can see in the network traffic my credentials are well exposed, so if some smart hacker is trying to sniff in the traffic, he can easily get hold of my credentials. Hence, all transactional websites always go for security certificate to prevent this kind of scenario. Hence, they acquire an **SSL Certificate** and install the same on the webserver, and hence users are forced to go through login process via "***HTTPS***". However, we could also use one action filter known as "[***RequireHttps***]" which makes sure that request is arriving is coming over a secure encrypted connection.

dbo.webpages_Membership [Data] | dbo.UserProfile [Data] | AccountModels.cs | Register.cshtml | AccountController.cs | _Layout.cshtml

MovieReview.Controllers.AccountController | LogOff()

```
        }

        //
        // POST: /Account/Login

        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        [RequireHttps]
        0 references
        public ActionResult Login(LoginModel model, string returnUrl)
        {
            if (ModelState.IsValid && WebSecurity.Login(model.UserName, model.Password, persistCookie: model.RememberMe))
            {
                return RedirectToLocal(returnUrl);
            }

            // If we got this far, something failed, redisplay form
            ModelState.AddModelError("", "The user name or password provided is incorrect.");
            return View(model);
        }
```

But, for this to work you need to install the SSL certificate on the server, otherwise it will return 500 error from the server. Hence, for now just undo that change.

Authorize:-

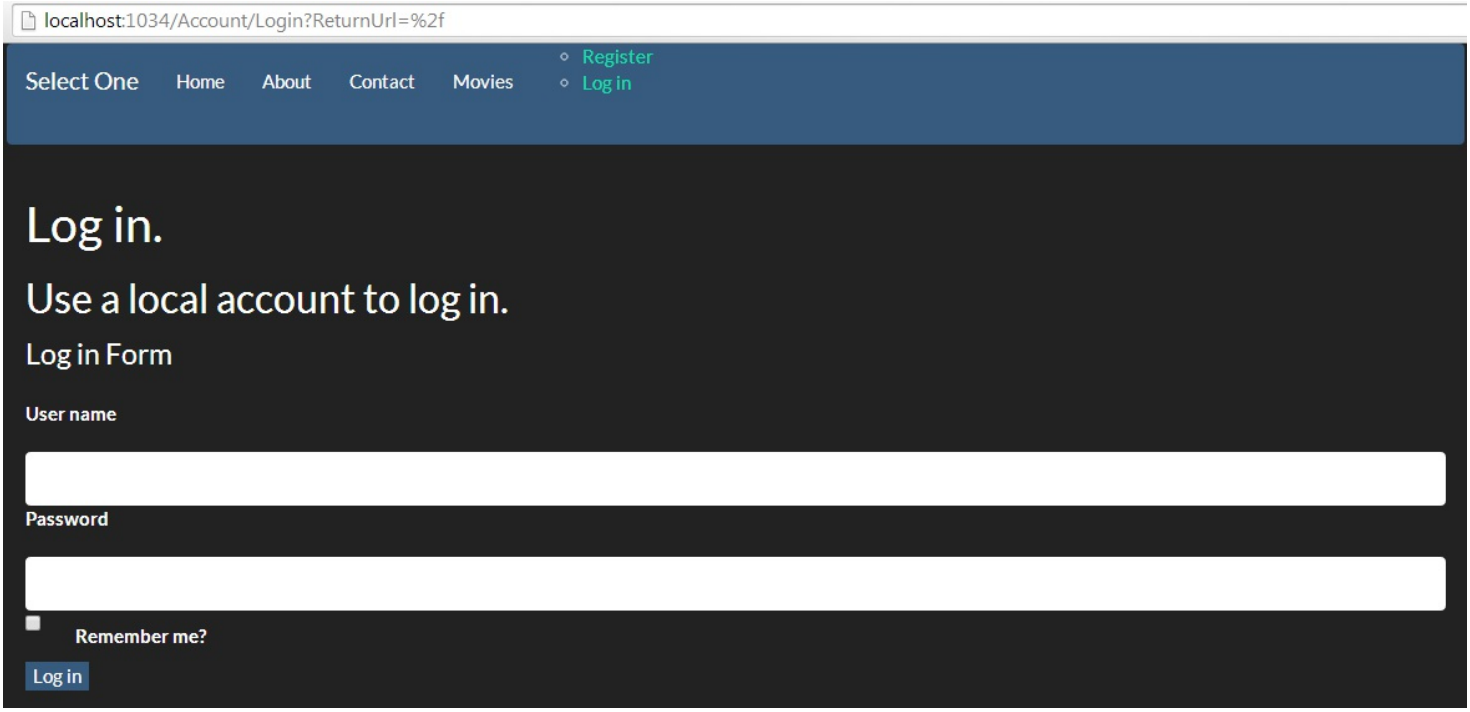
Now, let me go ahead and explain the Authorize attribute. Let's suppose that in

your application, you want to restrict few pages or few actions only to be accessed by authorized people, and then in that case we can use this action filter. We can either put the same before the **Controller** itself so that entire actions in that controller get restricted or to a particular action. Let, me show that. So, now what I am going to do is to restrict the **Index Action** in the **Home Controller**, so it will only be available when user logged in to the site.

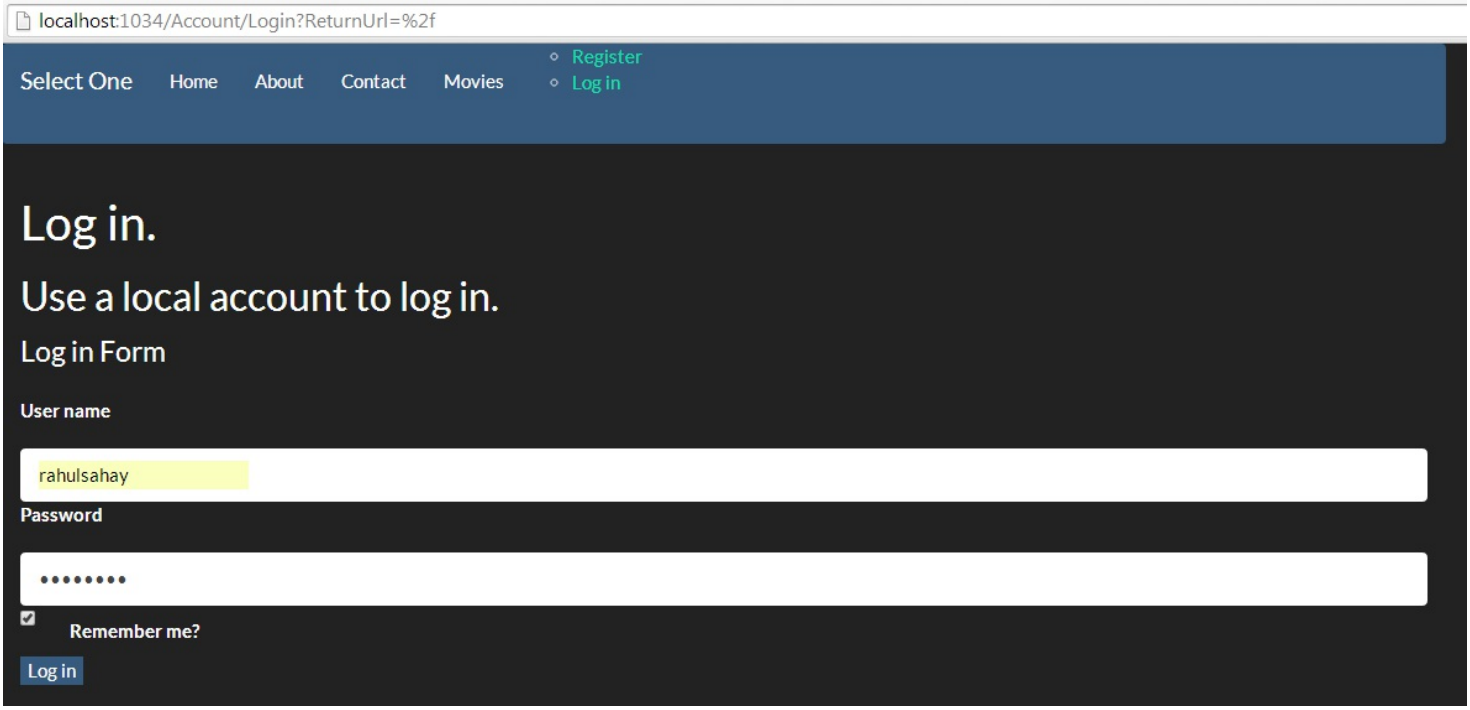
```
[Authorize]
        public ActionResult Index(string SearchQuery = null,int page=1)
        {

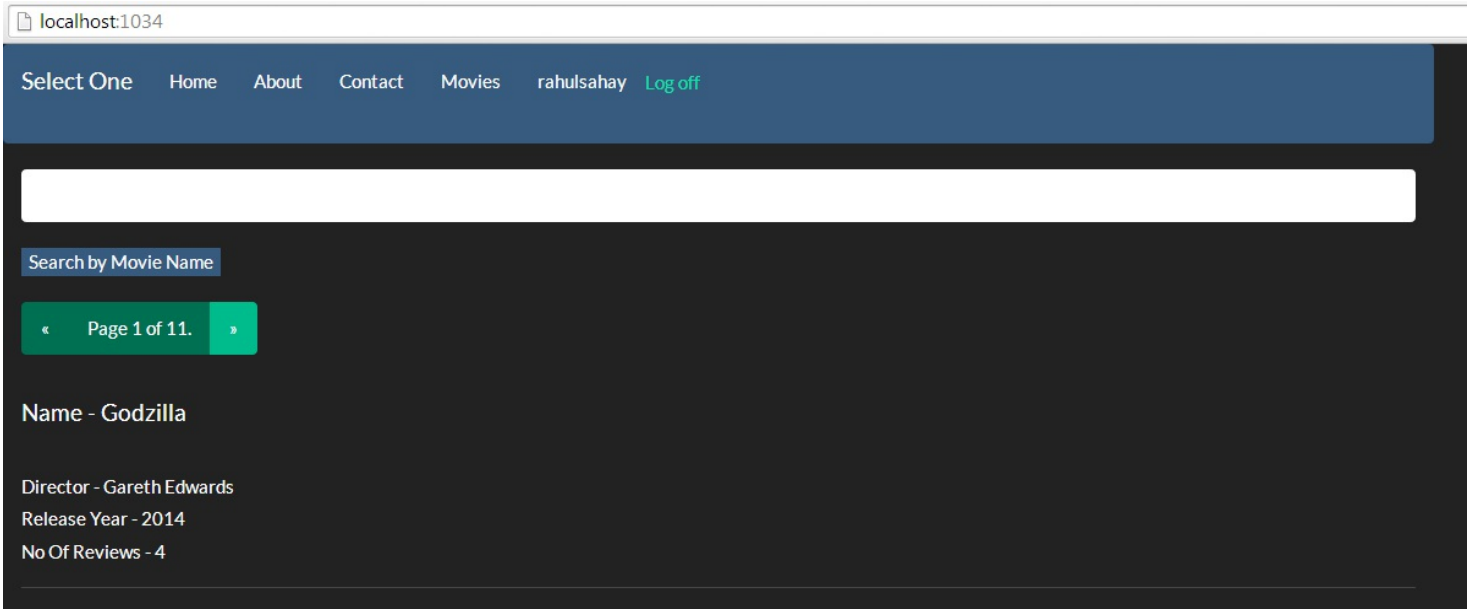
            var model = _movieDb.Movies
                        .OrderByDescending(m => m.reviews.Count())
                  .Where(m=>SearchQuery==null||m.MovieName.StartsWith(SearchQuery))
                        .Select(m => new MovieViewModel
                        {
                            Id = m.Id,
                            MovieName = m.MovieName,
                            DirectorName = m.DirectorName,
                            ReleaseYear = m.ReleaseYear,
                            NoOfReviews = m.reviews.Count()
                        }).ToPagedList(page,10);
            if (Request.IsAjaxRequest())
            {
                return PartialView("_movies", model);
            }
            return View(model);
        }
```

Now, what will happen here, when I try to navigate to the Home link, it will navigate me to the Login page 1st.



Now, when I provide my login credentials in there, then only I would be able to view the Home page as shown below.





So, as you can see in the above screen shot it navigated me to the required home page. Now, I can also use this authorize attribute with the user name or

user names separated by comma as shown below:-

```
[Authorize(Users= "rahulsahay,nivedita")]
        public ActionResult Index(string SearchQuery = null,int page=1)
        {

            var model = _movieDb.Movies
                        .OrderByDescending(m => m.reviews.Count())


          .Where(m=>SearchQuery==null||m.MovieName.StartsWith(SearchQuery))
                .Select(m => new MovieViewModel
                {
                    Id = m.Id,
                    MovieName = m.MovieName,
                    DirectorName = m.DirectorName,
                    ReleaseYear = m.ReleaseYear,
                    NoOfReviews = m.reviews.Count()
                }).ToPagedList(page,10);
    if (Request.IsAjaxRequest())
    {
        return PartialView("_movies", model);
    }
    return View(model);
}
```

And also I can do the filtering based on the roles. So, let's assume that you have certain roles created in your membership

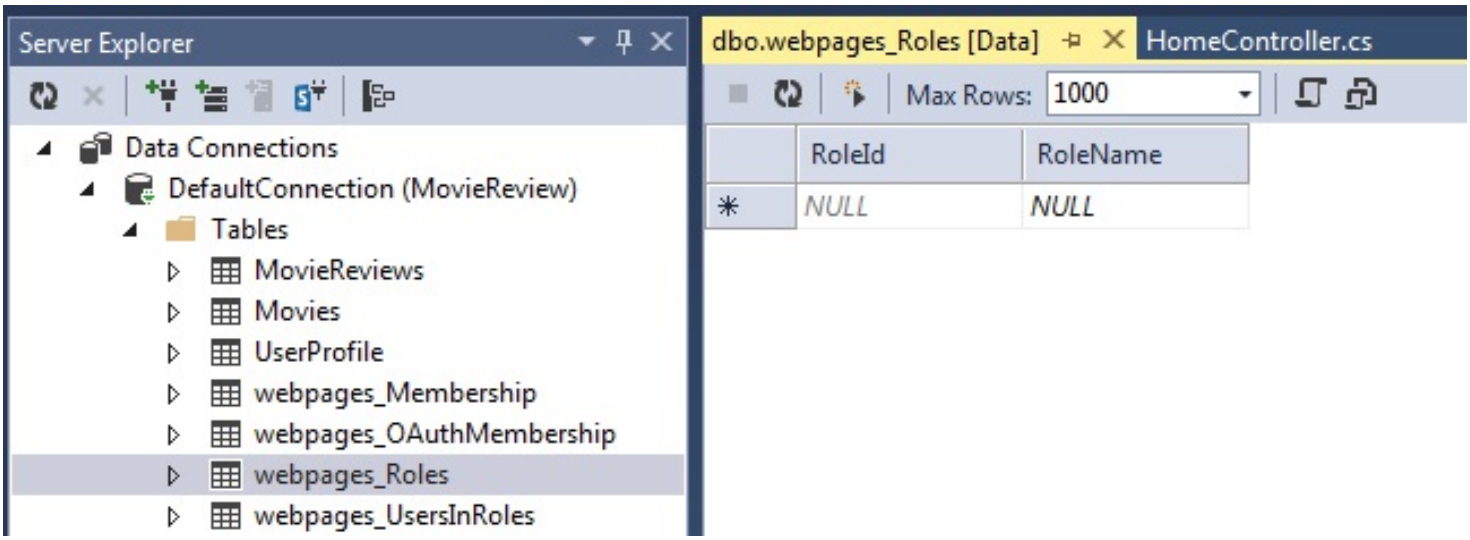
database like **Admin** and **SuperAdmin**, and then you can use them as shown below.

```
 [Authorize(Roles= "admin,superadmin")]
public ActionResult Index(string SearchQuery = null,int page=1)
{

    var model = _movieDb.Movies
                .OrderByDescending(m => m.reviews.Count())
          .Where(m=>SearchQuer==null||m.MovieName.StartsWith(SearchQuery))
                .Select(m => new MovieViewModel
                {
                    Id = m.Id,
                    MovieName = m.MovieName,
                    DirectorName = m.DirectorName,
                    ReleaseYear = m.ReleaseYear,
                    NoOfReviews = m.reviews.Count()
                }).ToPagedList(page,10);
    if (Request.IsAjaxRequest())
    {
        return PartialView("_movies", model);
    }
    return View(model);
}
```

Now, I have defined roles in there with the action filter, but we haven't created any roles yet as you can see the below

roles table is empty so far.



Seeding Membership:-

Now, in this section we need to see the way that how could we populate data in the **Users** and **Roles** table. So, basically having a database with users and roles predefined in the database is good during development. Now, there are several ways which you can use to insert

data into these tables like you could write simple SQL statements to insert records into these tables. But, we will be using membership **API** here to populate these data. So, now I am going to seed these data by writing a new method in the "***Configuration class***" as shown below:-

```
using System.Web.Security;

namespace MovieReview.Migrations
{
    using MovieReview.Models;
    using System;
    using System.Collections.Generic;
    using System.Data.Entity;
    using System.Data.Entity.Migrations;
    using System.Linq;
    using WebMatrix.WebData;

 internal  sealed  class
           Configuration:DbMigrationsConfiguration<MovieReview.Models.MovieDb>
    {
        public Configuration()
        {
            AutomaticMigrationsEnabled = true;
        }

        protected override void Seed(MovieReview.Models.MovieDb context)
        {
            //  This method will be called after migrating to the latest version.
```

```
            context.Movies.AddOrUpdate(r => r.MovieName,
                new Movie { MovieName = "Avatar", DirectorName = "James Cameron",
ReleaseYear = "2009" },
                new Movie { MovieName = "Titanic", DirectorName = "James Cameron",
ReleaseYear = "1997" },
                new Movie { MovieName = "Die Another Day", DirectorName = "Lee
Tamahori", ReleaseYear = "2002" },
                new Movie
                {
                    MovieName = "Godzilla",
                    DirectorName = "Gareth Edwards",
                    ReleaseYear = "2014",
                    reviews = new List<MovieReviews>{
                       new
MovieReviews{ReviewerRating=5,ReviewerComments="Excellent",ReviewerName="Rahul
Sahay"}
                    }
                });
            for (int i = 0; i < 100; i++)
            {
                context.Movies.AddOrUpdate(m => m.MovieName,
                new   Movie   {   MovieName   =   "Anony_Movie_"   +   i.ToString(),
DirectorName = "Anony_Director_" + i.ToString(), ReleaseYear = "2014" });
            }

            SeedUsersAndRoles();
        }

        private void SeedUsersAndRoles()
        {
            WebSecurity.InitializeDatabaseConnection("DefaultConnection",
"UserProfile", "UserId", "UserName", autoCreateTables: true);

            var roles = (SimpleRoleProvider) Roles.Provider;
            var membership = (SimpleMembershipProvider) Membership.Provider;

            //Apply all combinations
            if (!roles.RoleExists("Admin"))
            {
                    roles.CreateRole("Admin");
            }
            if (membership.GetUser("rahulsahay", false) == null)
            {
                membership.CreateUserAndAccount("rahulsahay", "password");
            }
```

```
            if (!roles.GetRolesForUser("rahulsahay").Contains("Admin"))
            {
                roles.AddUsersToRoles(new []{"rahulsahay"},new []{"Admin"});
            }
        }
    }
}
```

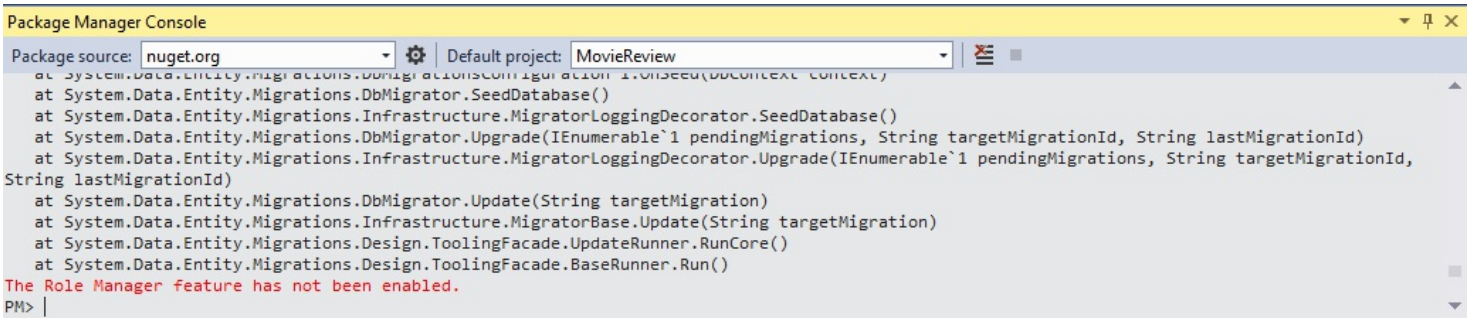
So, as you can see here, that it uses the call for "***WebSecurity.InitializeDatabaseConne*** just to make sure that everything is setup and schema is in place. Then, once the database is setup I can walk through **Current Role** provider and **Current Membership** provider by using the property provider on the roles class and the membership class. So, now I have access to two objects that I can use to create users or if user exists then see which role that user is in. and then below is few permutation and

combination which we could use.

So, the 1st thing is checking if there is a role called "***Admin***" if not then create one for me. 2nd thing checking that is there any user with the name "***rahulsahay***", if not then create one with the same name and password. And the last one is checking if the user "***rahulsahay***" is in **admin role**; if not then assign this role to the user. So, it's always a good idea to check whether the things exist before just directly inserting because seed method runs always whenever we update the database from the package manager console.

Now, with this code in place, I can go

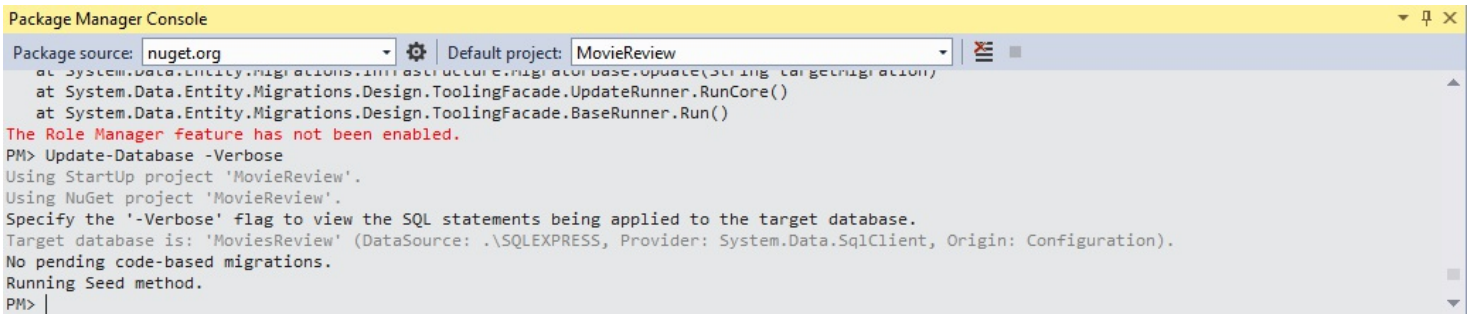
ahead and run the update command (Update-Database –Verbose) from the package manager console as shown below:-



So, as you can see in the above screen shot it threw me an error. Since, here we are running update database part of EF Migrations, hence it requires little bit of configuration in the "***web.config***" file to work. So, we need to add below snippet in the "***system.web***" section.

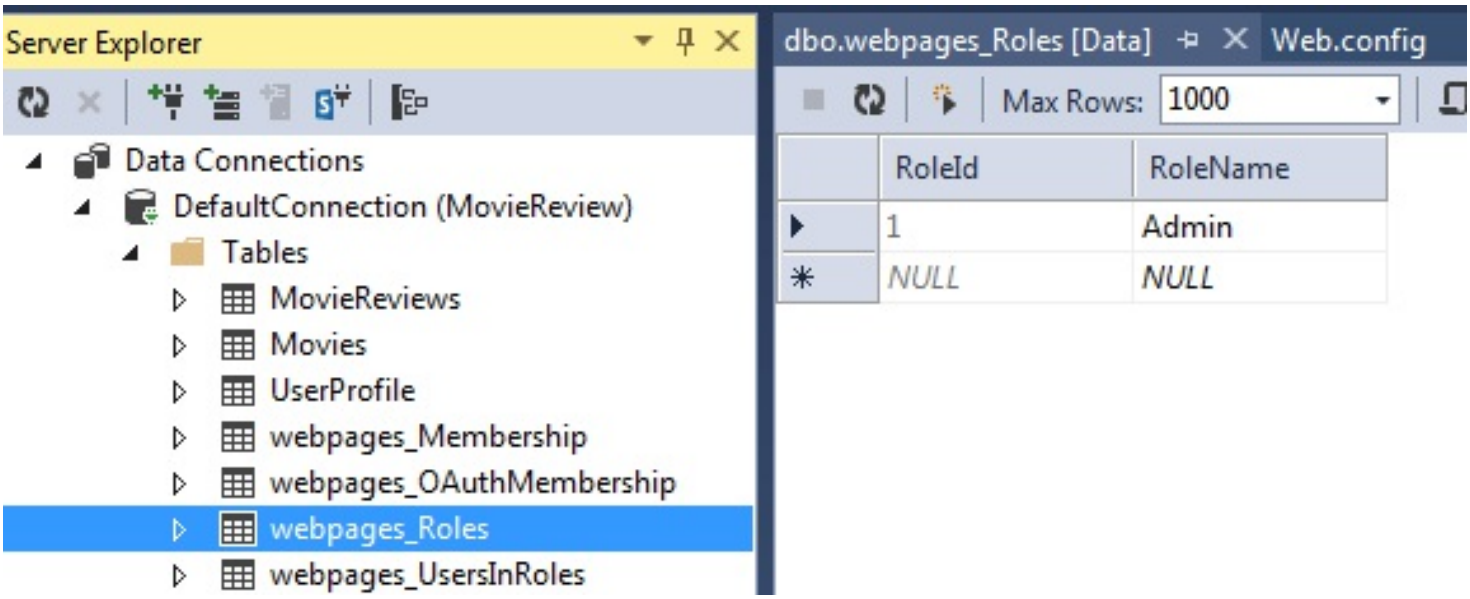
```
  <roleManager enabled="true" defaultProvider="simple">
    <providers>
      <clear/>
      <add name="simple" type="WebMatrix.WebData.SimpleRoleProvider,WebMatrix.WebData" />
    </providers>
  </roleManager>
  <membership defaultProvider="simple">
    <providers>
      <clear/>
      <add name="simple" type="WebMatrix.WebData.SimpleMembershipProvider,WebMatrix.WebData" />
    </providers>
  </membership>
</system.web>
```

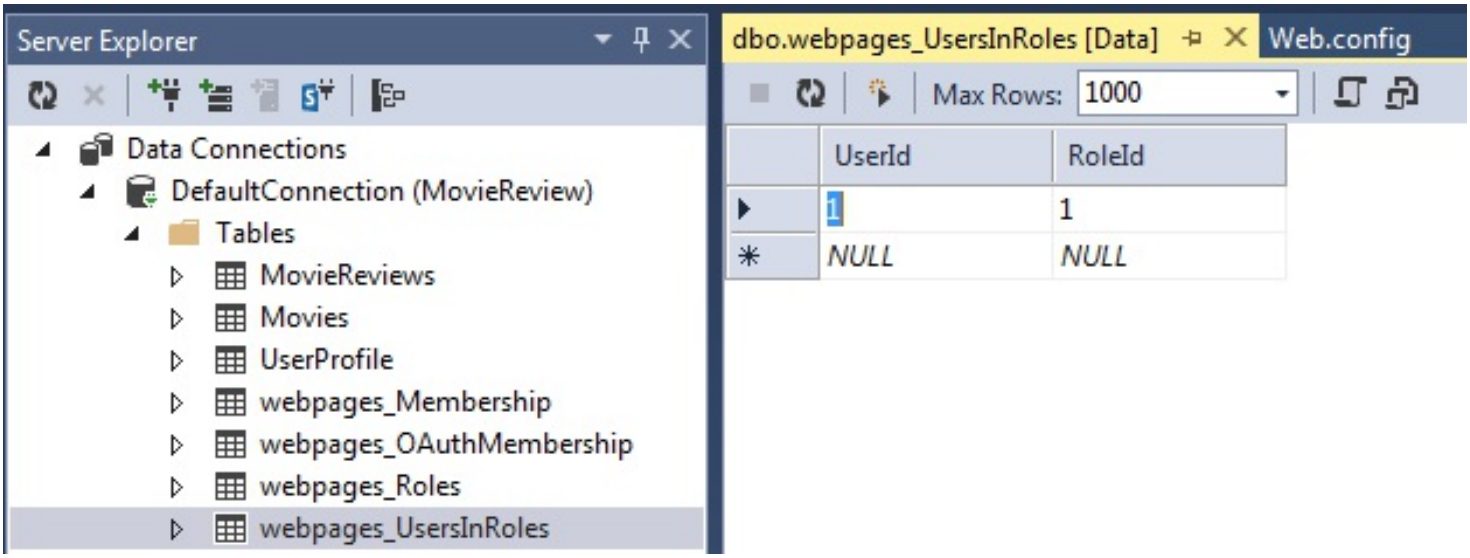
Now, with the above piece of code in place, we should be able to run this update command from the package manager console.



Now, let's verify whether the data has been migrated successfully or not. So, now, when I open the "***roles***" and "***userInRoles***" table, I can see below

results.





Now, let's remove the authorize attribute from the home controller and put the same before **Create Movie Action**. So,

now I want only authenticated users in admin role should be able to create new movie in the database. So, with the below piece of code, I should be able to achieve the same.

```
// GET: /Movies/Create
        [Authorize(Roles = "Admin")]
        public ActionResult Create()
        {
            return View();
        }

        //
        // POST: /Movies/Create

        [HttpPost]
        [Authorize(Roles = "Admin")]
        [ValidateAntiForgeryToken]
        public ActionResult Create(Movie movie)
        {
            if (ModelState.IsValid)
            {
                db.Movies.Add(movie);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            return View(movie);
        }
```

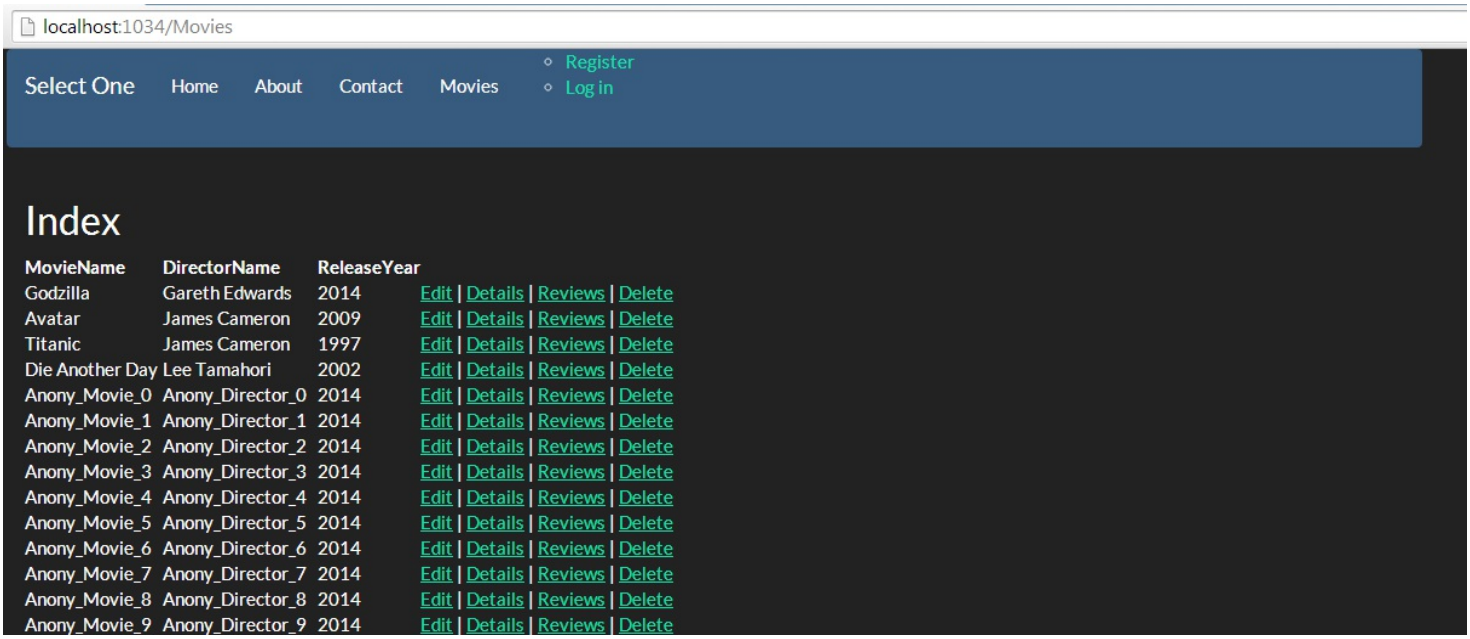
Now, I would also need to do one thing, like I don't want to show "***Create New***" link to the users who are not admin. So, to do the same, we need to change the code in Index view for movie as shown below:-

```
<p>
    @if (User.IsInRole("Admin")) {
    @Html.ActionLink("Create New", "Create")
    }
</p>
```

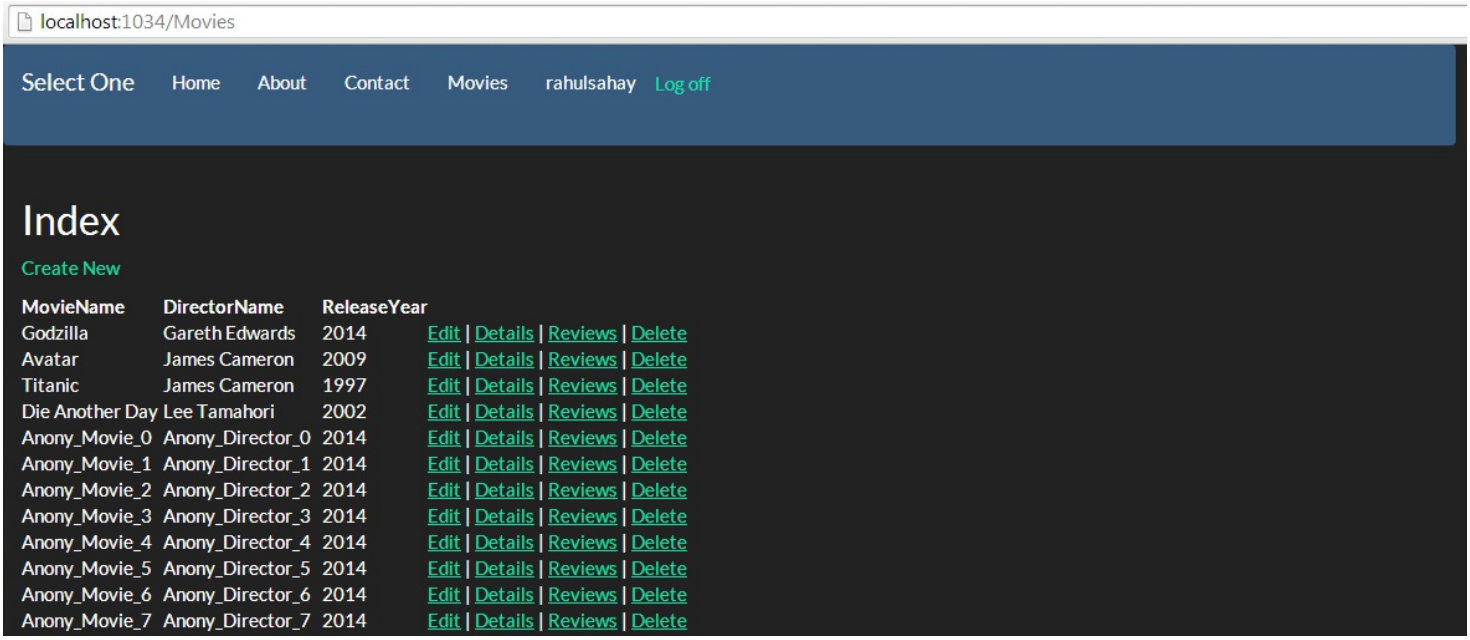
Now, with the above piece of code in place I should be able to go and test my application. So, just build the app once and refresh the page.

Now, when I click on movies link, it will display me the below result.

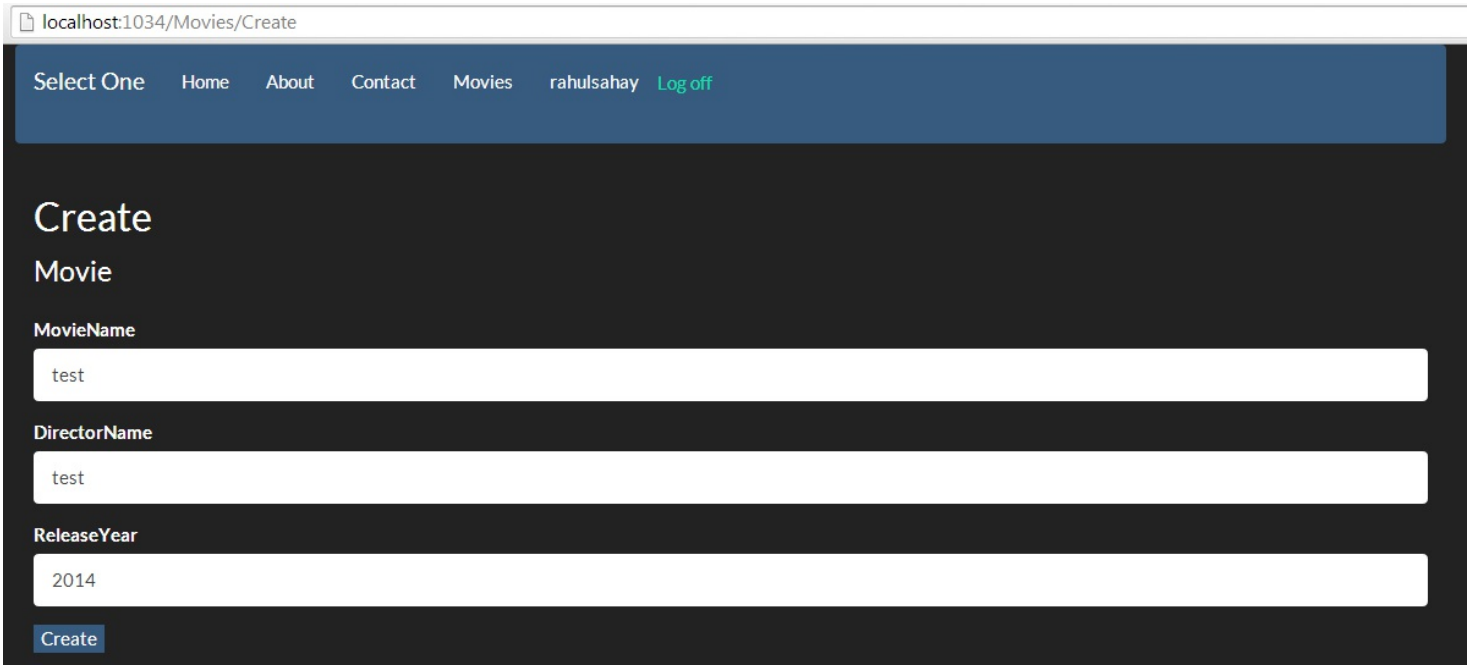
Notice, there is no "***Create New***" link there as I haven't logged in yet.

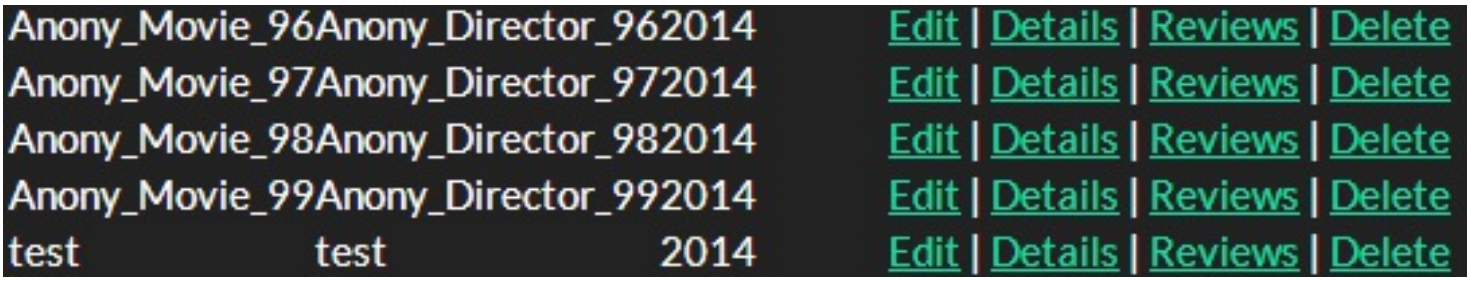


Now, when I login and then try to navigate to this page, I should be able to see the "***Create New***" link and create the movie as well.



And now, when I try to create new movie I should be able to create the movie as well as shown below.

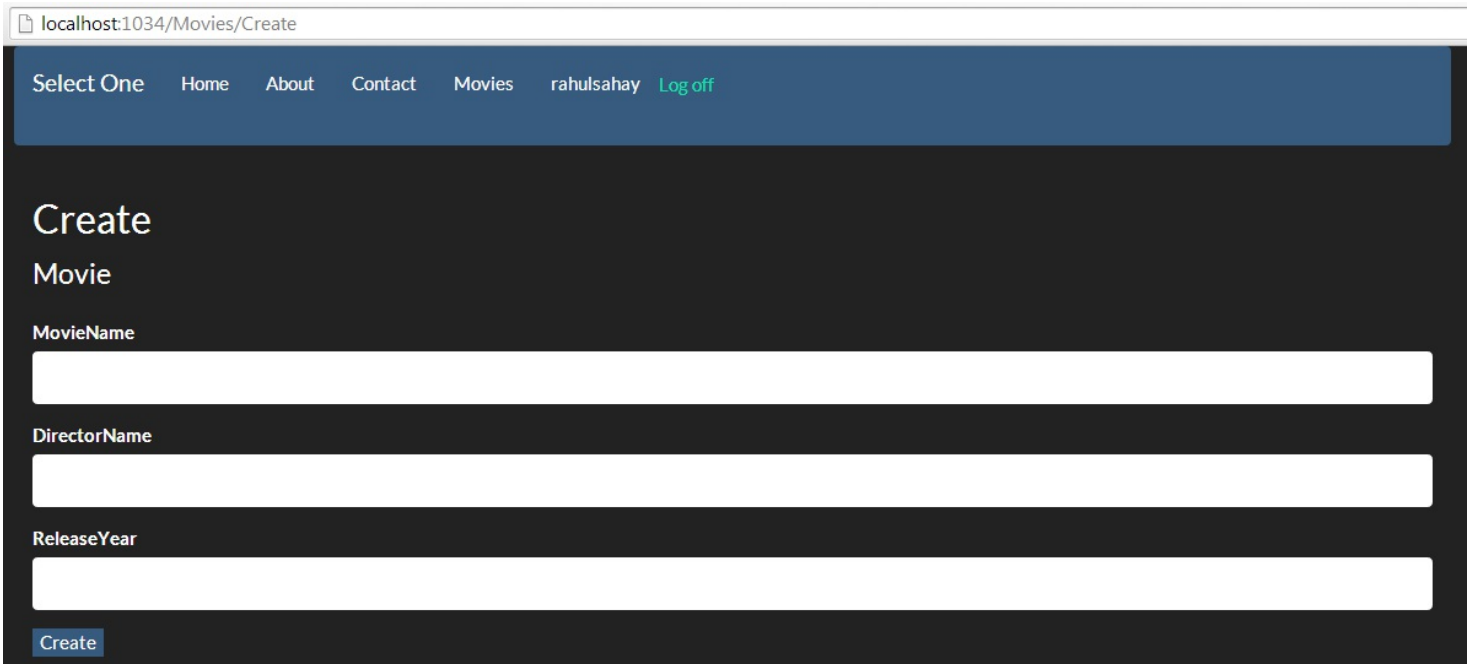




Cross Site Vulnerability:-

Now, in this section we will discuss some of the most common attacks tried by the hackers and that is known as "***Cross Site Request Forgery***" also known as "***CSRF***". **CSRF** is basically an attack that a malicious user can attempt against your site to get the admin access or to delete the data or damage the data. Now, let me explain the same by giving a short example using our application. Let's suppose that I am

malicious user and I don't have admin access and I want to create a movie in the site. So, what I would do. I would go ahead and make a replica of your application by just creating a simple HTML page and I would get these HTML Tags and associated values from the source code of the actual rendered page. Let me demonstrate that:-



So, as an admin user I would like to

navigate to this page, to create a new movie. But malicious user can't do that. So, he will design exactly the same page by keeping the same look and feel. Now, let me open a new Visual Studio project and create one empty project for the same.



Then I'll add a new Item just a regular HTML page as shown below. Now, in order to design similar kind of page, I would just know little bit of the HTML

that how the application looks like and based on that I would design the HTML. But, basically am interested inside the form tag which is getting generated when the actual page get rendered.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
</head>
<body>
    <form id="myForm"
style="display:none"action="http://localhost:1034/Movies/Create" method="post">

<input  class="text-box  single-line"  id="MovieName"  name="MovieName"  type="text"
value="Hacked :)" />

<input class="text-box single-line"id="DirectorName"   name="DirectorName"
type="text" value="Hacked :)" />

<input class="text-box single-line" id="ReleaseYear"  name="ReleaseYear"
type="text" value="Hacked :)" />

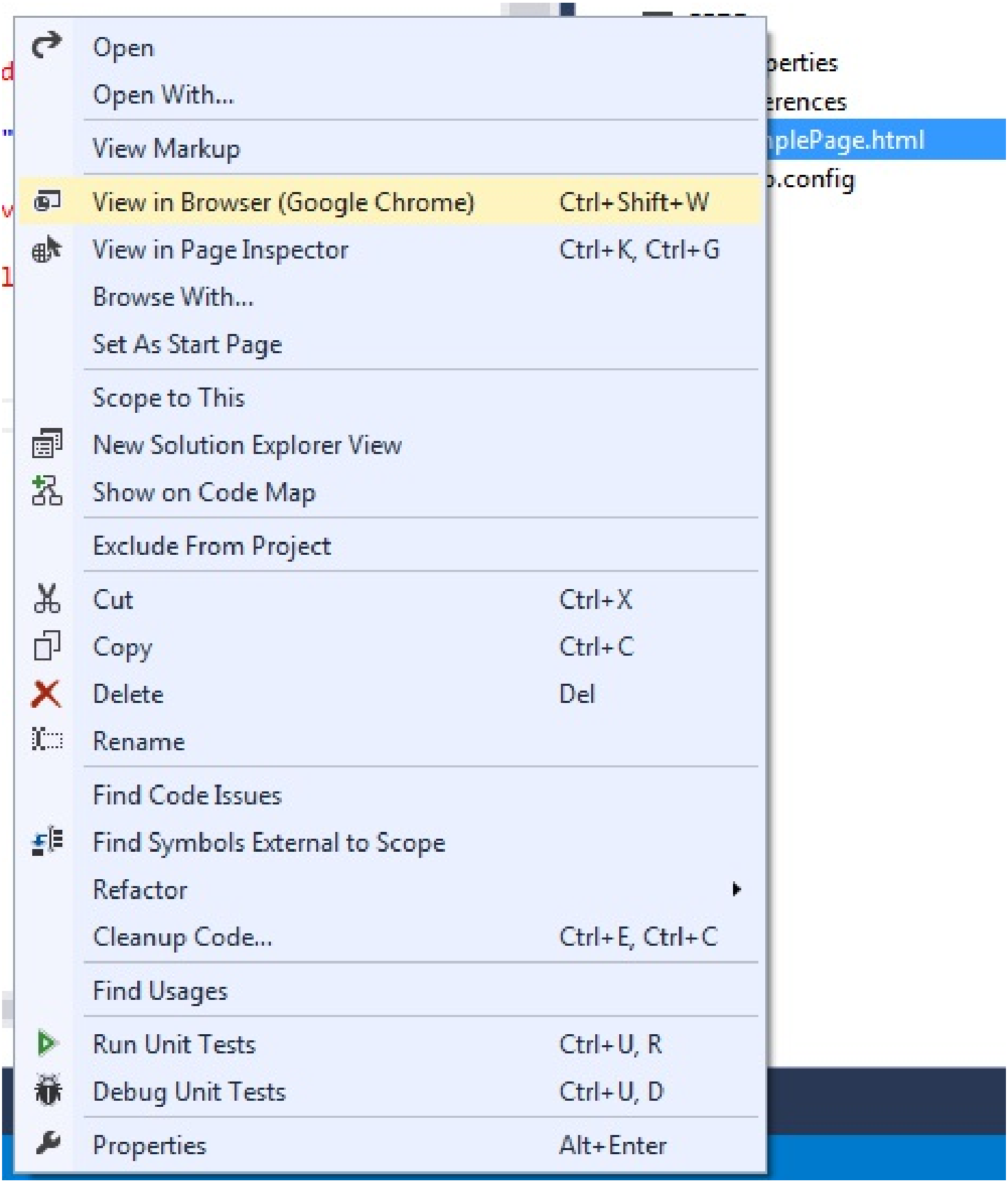

    </form>

    <script type="text/javascript">
        setTimeout(function() { window.myForm.submit(); }, 3000);
    </script>
</body>
</html>
```

So, if you see the html page, this is quite

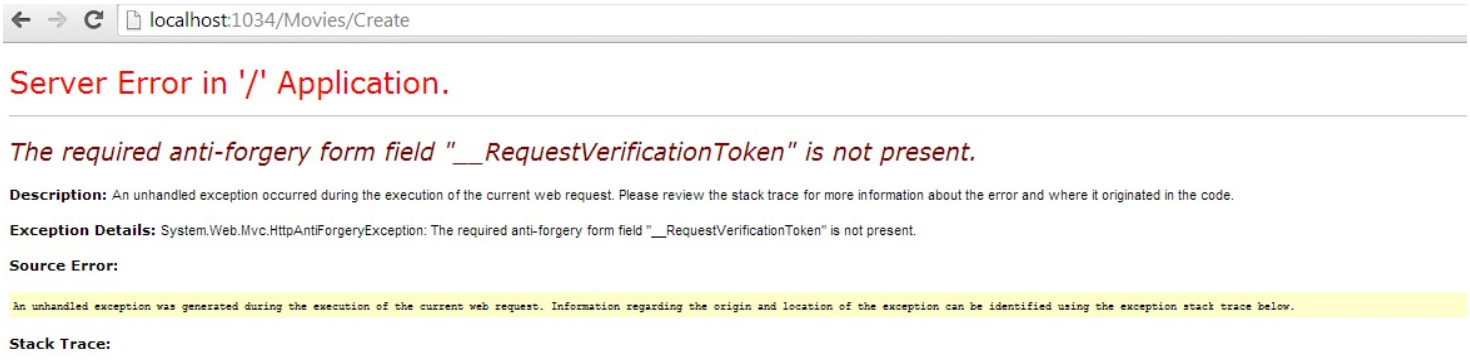
simple page, having three input tags with my dummy values inside and one point to notice here is the action item. If you notice the action item it is just redirecting you to the original site page, so actually it will take these inputs and submit in your application. Also, I have added one script there which will just automatically submit this form after 3000 millisecond delay.

Now, if simply open this html page by just right-clicking and view the same in browser, the form will automatically get submitted to my site and hence log in page just popped up.



Then, it will produce me the below

output.



It happened because in my create action I already enabled to prevent this action "[ValidateAntiForgeryToken]". Now, if remove this action filter and try the same again, then it will produce me a different picture. Let me show you that as well.



So, as you can see in the above screen

shot that malicious user injected his values inside the application. So, it happened because I as admin already logged in and hence it considered my browsing session and posted these values in my application. So, in actual scenario, what malicious user will do copy my newly hosted link and email to different admin users, who just clicked link by thinking some valuable page and if at that time he already has the session opened in a different window, then it will post the malicious data into the application. So, in order to prevent these kind of scenarios we can put the same attribute "[ValidateAntiForgeryToken]" before the action method. So, with this attribute in place, it will restrict any

CSRF activities as we seen above.

```
[HttpPost]
        [Authorize(Roles = "Admin")]
        [ValidateAntiForgeryToken]
        public ActionResult Create(Movie movie)
        {
            if (ModelState.IsValid)
            {
                db.Movies.Add(movie);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            return View(movie);
        }
```

OpenId and OAuth:-

Now, let's discuss another type of authentication which is very popular in

today's era that is "***OpenId and OAuth***". These two technologies let your users authenticate themselves without you managing the passwords. This is very beneficial and efficient. Like people who already have some accounts resisted with some of the popular sites like **Google, Microsoft, Facebook** etc, then user doesn't have to get registered themselves with your membership site. Rather they can get authenticated themselves by any of these third party site and land in your application. MVC 4 and above has this built in support for Open Id and OAuth. Now, in order to learn more about these two technologies you can either refer (http://openid.net/, http://oauth.net/).

Here in our application it's very easy to enable these settings. So, inside this project we can see in the global.asax file, there is a call for "AuthConfig.RegisterAuth();" which lands you in **App_Start** folder in "***AuthConfig.cs***" file.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Web.WebPages.OAuth;
using MovieReview.Models;

namespace MovieReview
{
    public static class AuthConfig
    {
        public static void RegisterAuth()
        {
            // To let users of this site log in using their accounts from other
sites such as Microsoft, Facebook, and Twitter,
            //   you   must   update   this   site.   For   more   information   visit
http://go.microsoft.com/fwlink/?LinkID=252166

            //OAuthWebSecurity.RegisterMicrosoftClient(
            //    clientId: "",
            //    clientSecret: "");

            //OAuthWebSecurity.RegisterTwitterClient(
            //    consumerKey: "",
            //    consumerSecret: "");

            //OAuthWebSecurity.RegisterFacebookClient(
            //    appId: "",
            //    appSecret: "");

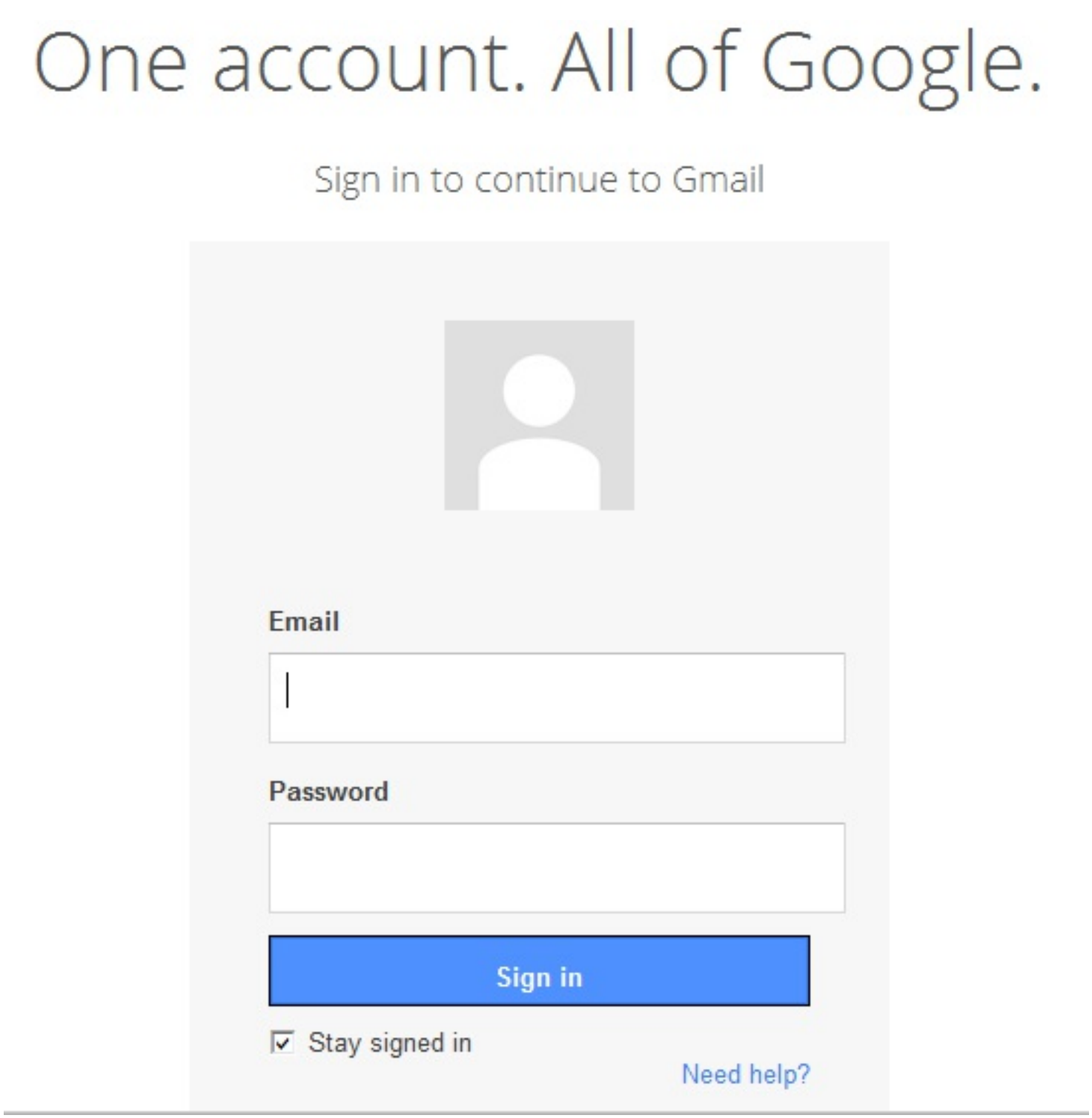
            //OAuthWebSecurity.RegisterGoogleClient();
        }
    }
}
```

Now, if you see the above code, currenty all the settings are commented. But, we could go ahead and enable all the

settings. If you notice closely, you will come to know that all of these settings except last one which is for **google**, require **Id** and **Secret** key. So, for getting that you need to register your site with their site and they will issue an Id and key for the same. But, for demonstration perspective google client is ok for me. So, just I need to uncomment the last setting, build the application and then refresh the login page as shown below.

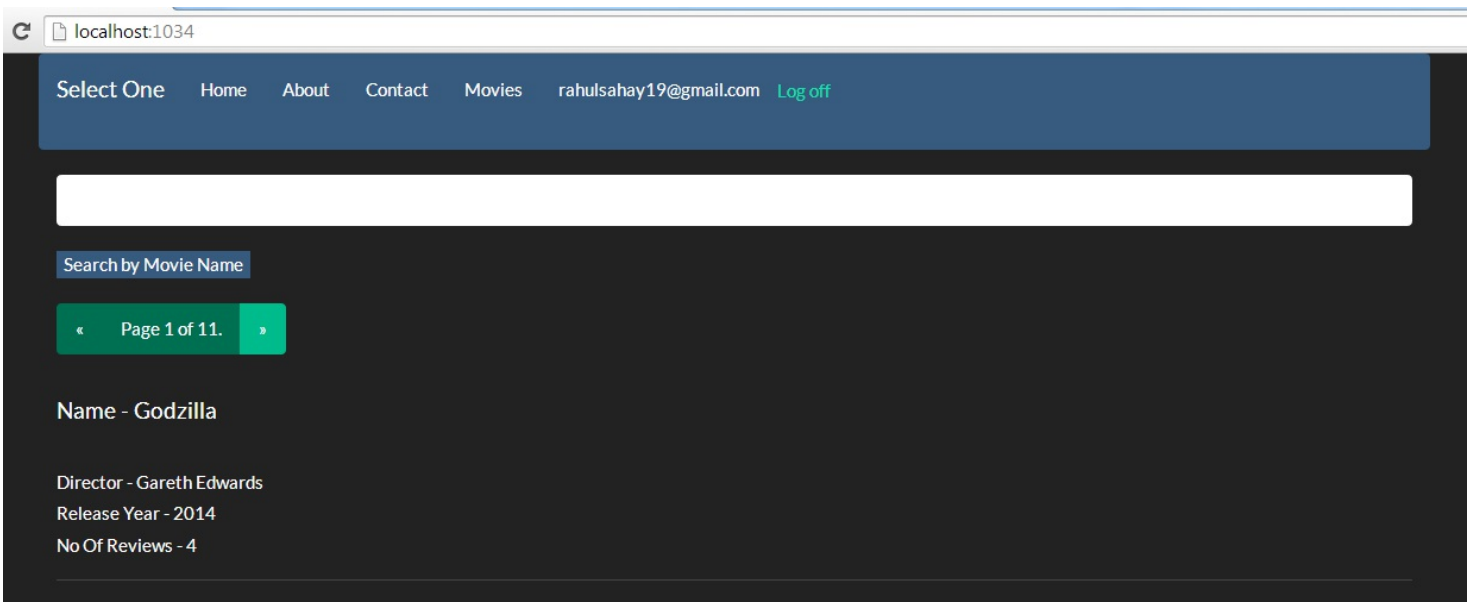


Now, when you click the Google link as shown above in the screen shot, it will take you the google link for authentication.



Now, here when login with your user name and password it will take you to our application after successful

authentication.



Summary:-

So, in this chapter we have seen security aspects involved in our MVC application. We see how to use Windows Authentication, Forms Authentication. We also saw how to set up roles and membership and take

ownership of some important tables like User Profile table. We have also seen WebSecurity methods in action. We also covered that how to prevent CSRF attacks if attempted. And last but not the least we have seen another kind of authentication that is OpenId and OAuth.

Chapter 8: Caching - ASP.Net Terminology

WHAT DO you find in this CHAPTER?

⇒ Introduction

⇒ Caching

⇒ Cache Profiles

⇒ Localization

⇒ Resource Files

⇒ Diagnostics

⇒ ELMAH

⇒ Summary

Introduction:-

Hi, folks. I hope you guys have been

enjoying the journey to learn MVC by doing End to End Development. Now, in this chapter we are going to focus on the common routines of ASP.NET Framework. Here, in this case we will be focusing some of the most common ASP.NET techniques like Caching to improve the performance of the application, Localization techniques to implement Globalization and Localization in the application and digging inside the application by using some Diagnostics settings.

Caching:-

Now, in this section we'll begin our

discussion with "***[OutputCache]***" action filter. So, Output caching basically allows you to store the output of a particular controller in memory. So, after that any future request coming for the same action in that controller will be returned from the cached result. Hence, in this case it avoids executing any of the code inside the action. So, now since we are executing less code and performing fewer actions, hence it improves the performance of the website dramatically. However, caching is easy to enable by using "***[OutputCache]***" filter, but you have to very careful when and where caching should be applied. Because, in many cases it may return you erroneous output and I'll explain those scenarios in

some time. Now, while using caching, variety of options you could specify in there like "***Duration***", "***VaryByParam***" etc. so, if you are familiar with ASP.NET Caching, then these terms won't be new for you. Now, let me explain the same with an example.

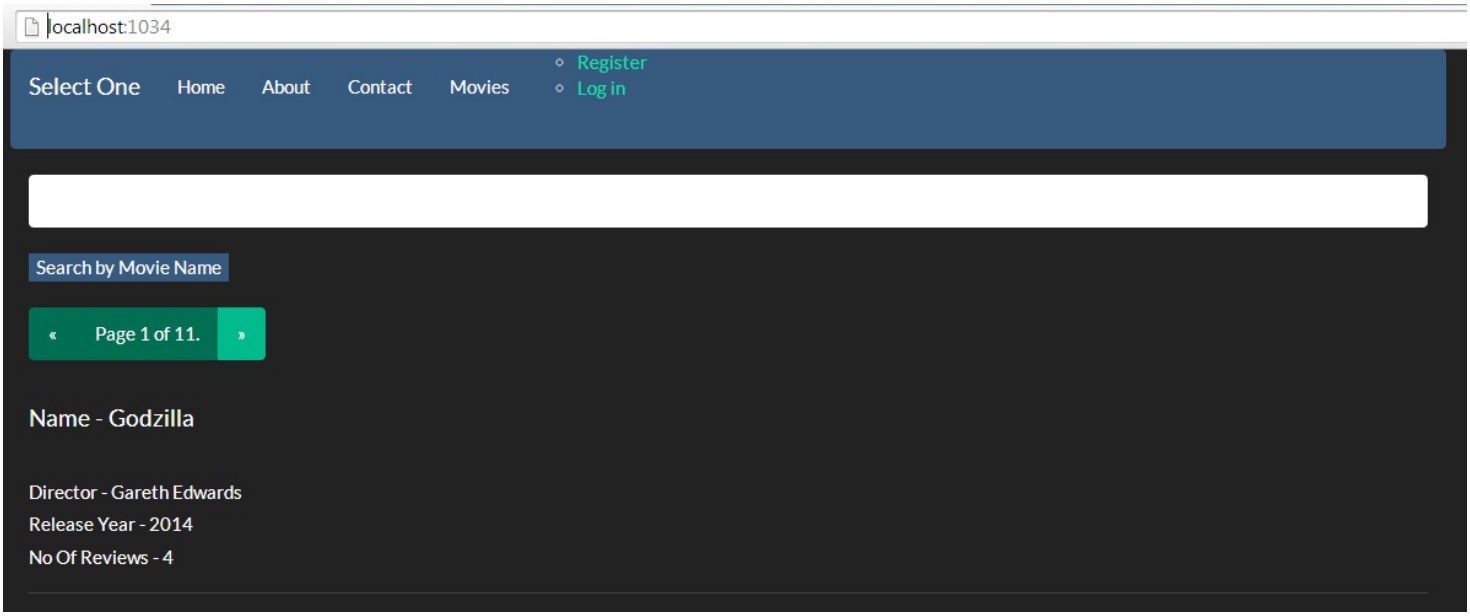
```
[OutputCache(Duration = 60)]
        public ActionResult Index(string SearchQuery = null,int page=1)
        {

            var model = _movieDb.Movies
                        .OrderByDescending(m => m.reviews.Count())
                  .Where(m=>SearchQuery==null||m.MovieName.StartsWith(SearchQuery))
                        .Select(m => new MovieViewModel
                        {
                            Id = m.Id,
                            MovieName = m.MovieName,
                            DirectorName = m.DirectorName,
                            ReleaseYear = m.ReleaseYear,
                            NoOfReviews = m.reviews.Count()
                        }).ToPagedList(page,10);
            if (Request.IsAjaxRequest())
            {
                return PartialView("_movies", model);
            }
            return View(model);
        }
```

Now, if you can see in the above snippet I have added "***OutputCache***" attribute on the index action of the controller. Since this is the home page of the application; hence, it receive highest traffic. So, it would be good idea to implement caching here to boost the performance. Now, in order to explain, I run the same in the debug mode, so what will happen here; at the 1st load, it will come inside the action and execute all the code written and then returns the result, then in the 2nd consecutive pass, results will get loaded from the cache and hence, it won't the hit controller's index action under that duration.

```
[OutputCache(Duration = 60)]
1 reference | 0/1 passing
public ActionResult Index(string SearchQuery = null,int page=1)
{

    var model = _movieDb.Movies
                .OrderByDescending(m => m.reviews.Count())
                .Where(m => SearchQuery == null || m.MovieName.StartsWith(SearchQuery))
                .Select(m => new MovieViewModel
                {
                    Id = m.Id,
                    MovieName = m.MovieName,
                    DirectorName = m.DirectorName,
                    ReleaseYear = m.ReleaseYear,
                    NoOfReviews = m.reviews.Count()
                }).ToPagedList(page,10);
    if (Request.IsAjaxRequest())
    {
        return PartialView("_movies", model);
    }
    return View(model);
}
```



Now, notice, I have the debugger on, but when I refresh the page, it won't hit the break point. So, it loaded the results

from the cache itself. So, for 60 seconds what I have specified in the parameter, it won't load the content from the server. Now, in addition of duration, there are other settings which you can also set here. So, please note that these settings are inherited from ASP.Net Caching, so it's not exclusive for MVC framework. Now, one of the very famous settings is "***VaryByParam***". So, it can be defined three ways as shown below:-

Vary by "*" means to cache for every set of parameters.- Now, this is the setting which is normally used because you don't want to return the same cached response for different parameters like query strings.

Vary by "none" to always cache the same content.

Vary by "name" to cache for every value passed in the parameter. Now, here what you can do; you can pass multiple parameters separated by semicolon.

So, let's suppose if I say **vary by param** for my "***SearchFilter***" and then do a search with the value "***Avatar***"; then it will be saved as 1st search result in cache and then then do a search with "***Titanic***" will be saved as 2nd cached result. So, for every different name it will save a different cache result. And, then if user searches "***Titanic***" again,

then it will be returned from the cached result, it won't come from the server.

Now, apart from these, we can also set "***Location***" like Cache on **Server, Client or Client and Server**. So, the default value in this case is anywhere means it will cache on the server or on the client. However, we can be specific with cache locations saying that it should be only stored on the server or on the client or on the proxy servers.

We can also specify "***VaryByHeader***". So, this setting allows you to vary the cache based on specific **HTTP Header** like "***Accept-Language***". It means we wouldn't want to return a response in

Spanish if it is being requested in English.

Then we have "***VaryByCustom***". Now, this setting will override a method inside "***global.asax***" file. So, inside here we can build our own custom caching mechanism telling ASP.Net what to cache.

Last but not the least is setting up "***SqlDependency***". So, basically a SQL Dependency tells that it cache responses until the data inside the SQL table changes. Now, all these settings are greatly explained in MSDN (http://msdn.microsoft.com/enus/library/v
So, you can do deep dive for this topic

by referring the article on MSDN. However, I'll do a demo on some of these topics which you will need or using most of the time. Now, the snippet which I have shown earlier won't work with paging because in there I have the parameter set as "int page=1". So, for every different page, it will hit the server, and then pull the details from the server. So, if you go back to page no. which you have searched then, it will not hit the index action. Now, let's suppose you have bookmarked few links and searching the same like ***http://localhost:1034/?SearchQuery=godzilla***, and then in that case also "***VaryByParam***" will work well. So, here the parameter is

"***SearchQuery***".

Now, let's suppose we would like to set caching location. Now, to do the same I need to bring in the namespace "***System.Web.UI***". So, below is the snippet which is telling that I have set the Cache Location on the server.

```
[OutputCache(Duration = 60,Location = OutputCacheLocation.Server)]
      public ActionResult Index(string SearchQuery = null,int page=1)
      {

          var model = _movieDb.Movies
                      .OrderByDescending(m => m.reviews.Count())
                .Where(m=>SearchQuery==null||m.MovieName.StartsWith(SearchQuery))
                      .Select(m => new MovieViewModel
                      {
                          Id = m.Id,
                          MovieName = m.MovieName,
                          DirectorName = m.DirectorName,
                          ReleaseYear = m.ReleaseYear,
                          NoOfReviews = m.reviews.Count()
                      }).ToPagedList(page,10);
          if (Request.IsAjaxRequest())
          {
              return PartialView("_movies", model);
          }
          return View(model);
      }
```

So, here ASP.Net will send instructions to the browser that response won't be cached in client location. So, for the response, browser will always have to come back to server and then check. Now, from the server side, ASP.Net will just return the response from memory if it has cached response. So, visually it doesn't make any changes means you won't realize any changes to the application, but only the storing mechanism now changed as server only.

Cache Profiles:-

Now, associated to these setting there is one more concept called

“***CacheProfiles***”. Let's suppose a scenario, that you have multiple pages in your application and each page you have set certain duration, but during the course of development, or might be in production, you want to increase or decrease the cache duration, then in that case it would be difficult to go ahead one by one in all the pages and change the duration. So, in order to overcome this scenario, ASP.Net has provided us “***CacheProfiles***” setting.

Cache profile setting can be done from “***web.config***” file. So, here we can have different profiles with different durations. So, on each of the controller's action, rather than specifying duration

we can specify profiles. So, idea behind this, this can be administered from the web.config file. So, below is the setting which you can specify in the web.config file under "***<system.web>***" section.

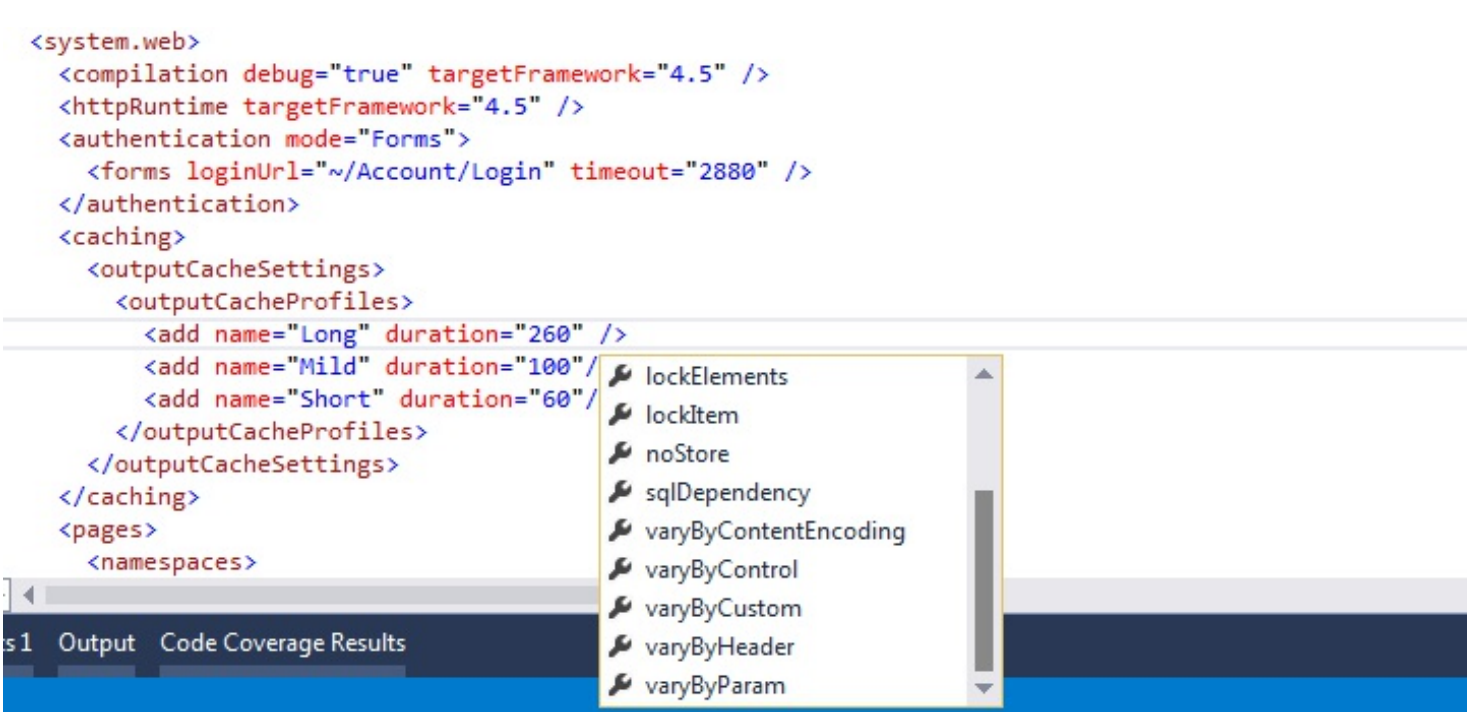
```
<caching>
      <outputCacheSettings>
        <outputCacheProfiles>
          <add name="Long" duration="260"/>
          <add name="Mild" duration="100"/>
          <add name="Short" duration="60"/>
        </outputCacheProfiles>
      </outputCacheSettings>
    </caching>
```

Now, from the controller side, you just need to specify like shown below. So, with the below piece of code in place, now it will load the duration what we

have specified in the web.config file. So, going ahead if you ever want to **increase/decrease** the time, you won't have to compile the solution again.

```
[OutputCache(CacheProfile = "Mild",Location = OutputCacheLocation.Server)]
        public ActionResult Index(string SearchQuery = null,int page=1)
        {

            var model = _movieDb.Movies
                        .OrderByDescending(m => m.reviews.Count())
                    .Where(m=>SearchQuery==null||m.MovieName.StartsWith(SearchQuery))
                        .Select(m => new MovieViewModel
                        {
                            Id = m.Id,
                            MovieName = m.MovieName,
                            DirectorName = m.DirectorName,
                            ReleaseYear = m.ReleaseYear,
                            NoOfReviews = m.reviews.Count()
                        }).ToPagedList(page,10);
            if (Request.IsAjaxRequest())
            {
                return PartialView("_movies", model);
            }
            return View(model);
        }
```



Also apart from Cache profiles, there is variety of settings which you can manage from the web.config file as shown above in the screen shot.

Localization:-

Now, the next topic which I would like to talk about is Localization. So, in this

case we'll see that how to localize the application. So, based on different cultures, application decides this localization behavior. So, there is couple of settings which decides this. Now, the first setting is "Thread.CurrentCulture" property. So, this property decided how to format the strings. For example, currency; means is it going to be rendered as Rs, Dollar or Euro, similarly based on this setting Time format for that particular zone is also decided. Another setting is "Thread.CurrentUICulture". So, this setting decides how the run time loads the resource. So, obviously these settings can be set via our own custom code like let user choose what language

they are interested in from the drop down. So, this is the manual case. Or, this can be done automatically as well. In this case we let ASP.NET handle this for me. In this case, ASP.Net can set cultures based on the HTTP Headers value, like “Accept-Language”. Now, in order to make this automatic, we need to add one “globalization” section in the web.config file as shown below.

```
<globalization culture="auto" uiCulture="auto"/>
```

Here, you could see that I have set the “***Culture***” and “***UICulture***” to auto. Now, let me do little demo with my application to explain the same.

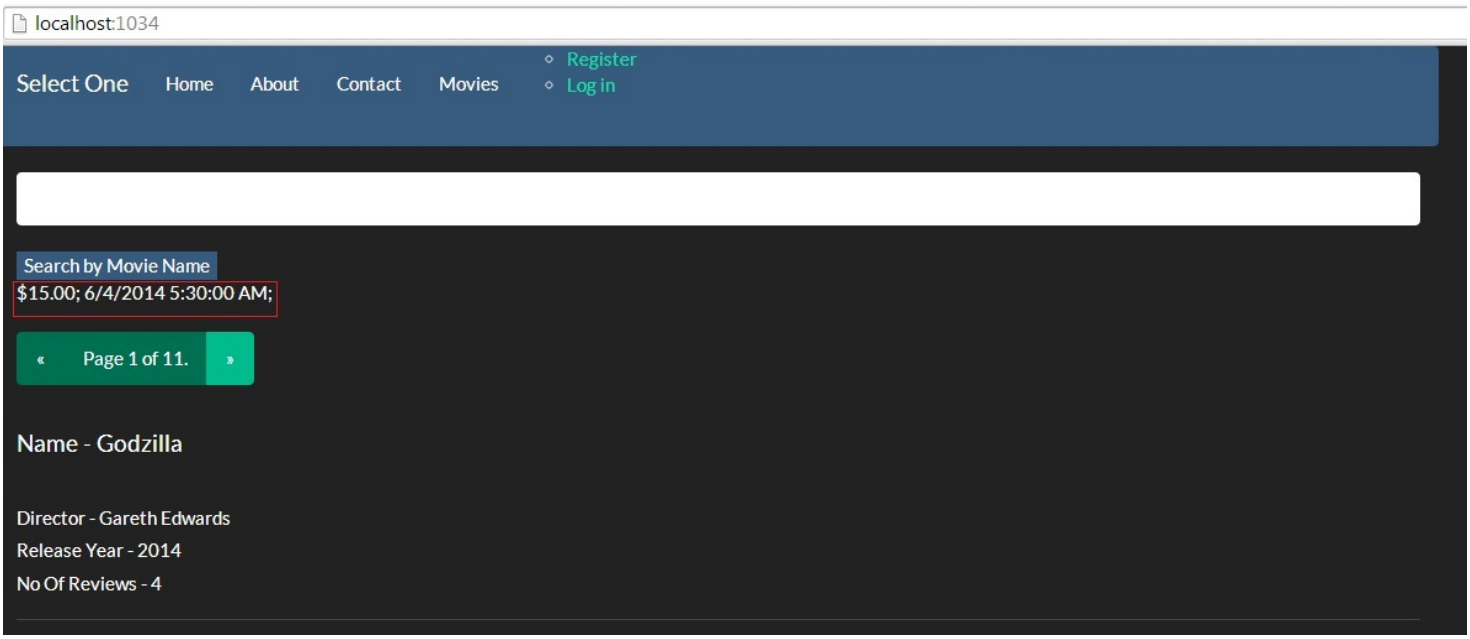
```
@{
    var amnt = 15.0m;
    var localDate = new DateTime(2014, 6, 4);
}

<div>

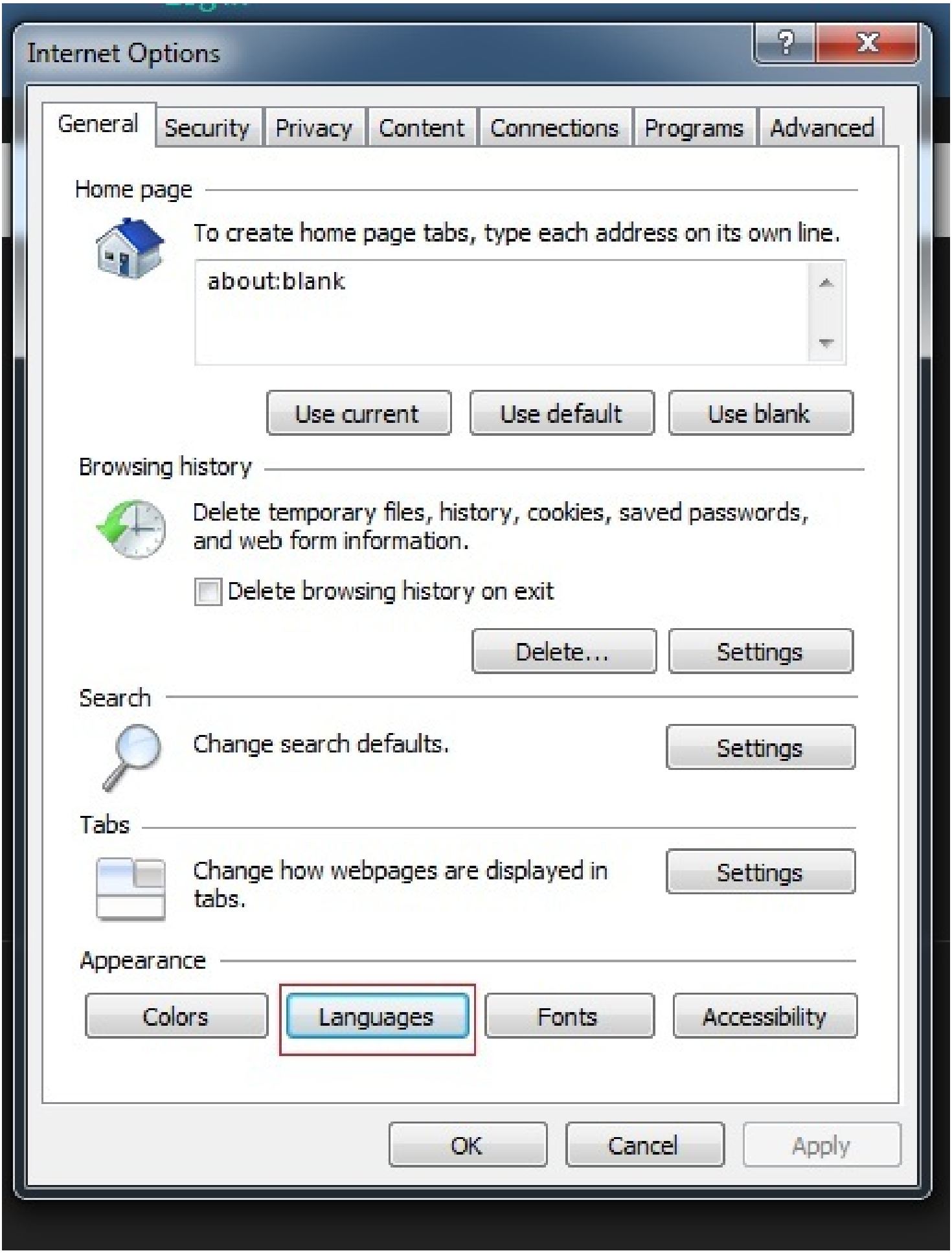
    @amnt.ToString("c");
    @localDate.ToLocalTime();
</div>
```

So, as you can see above, I have added a code block in which I have added an "***amnt***" variable which is taking decimal value and date in our regular format. Next, below in the div section I am outputting the same. So, in the div, you notice that am outputting the value as "***c***" parameter, which means please take this input and format the same as

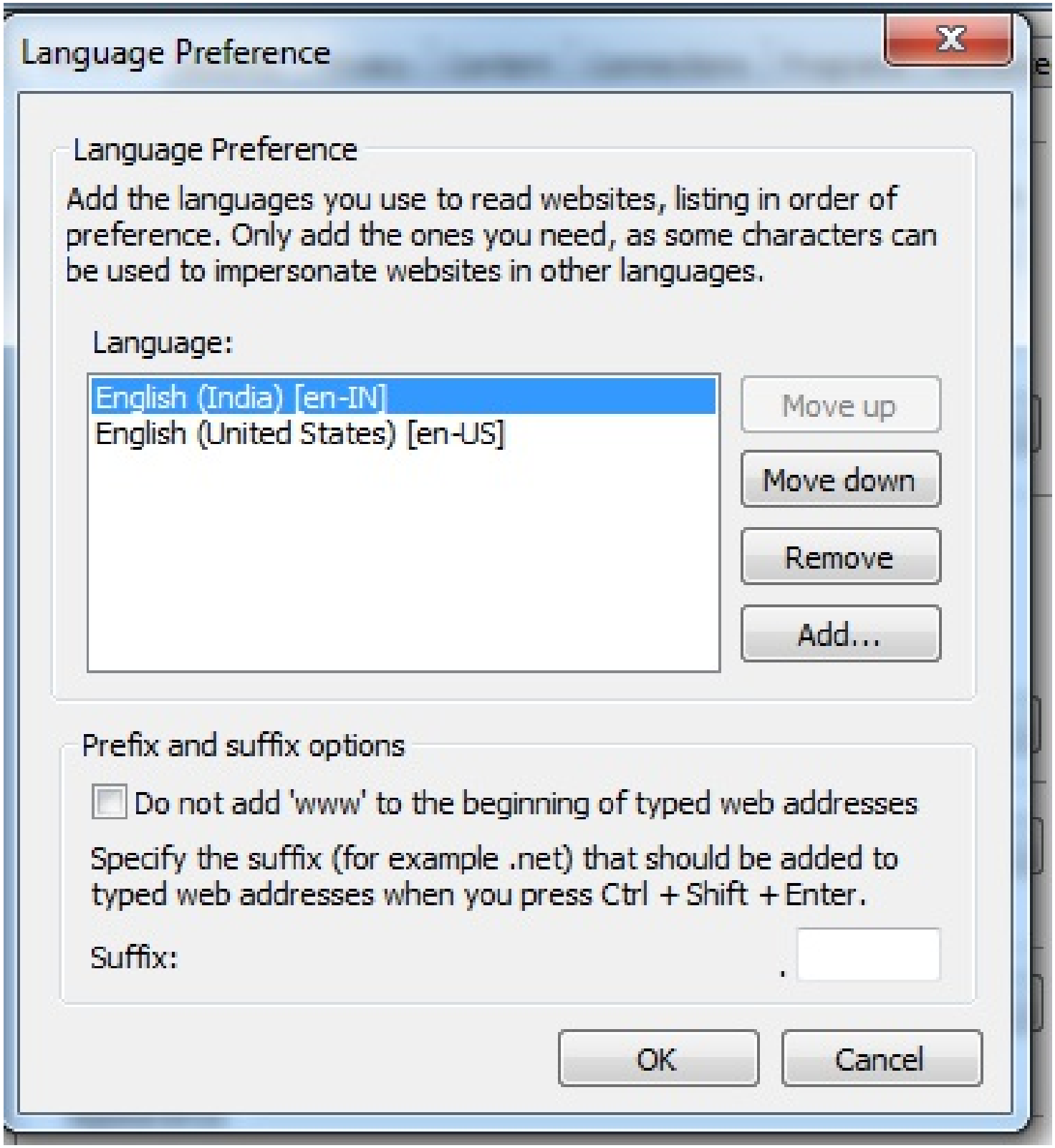
currency and then I am outputting the local time. Now, with the above change in place, when I save and refresh, then it will print the below output.



Now, since my default culture in here in this machine is set to "***United States***", hence it produced me the "***$***" symbol and then the local time. Now, from the IE settings, I can go ahead and change the language settings as shown below:-

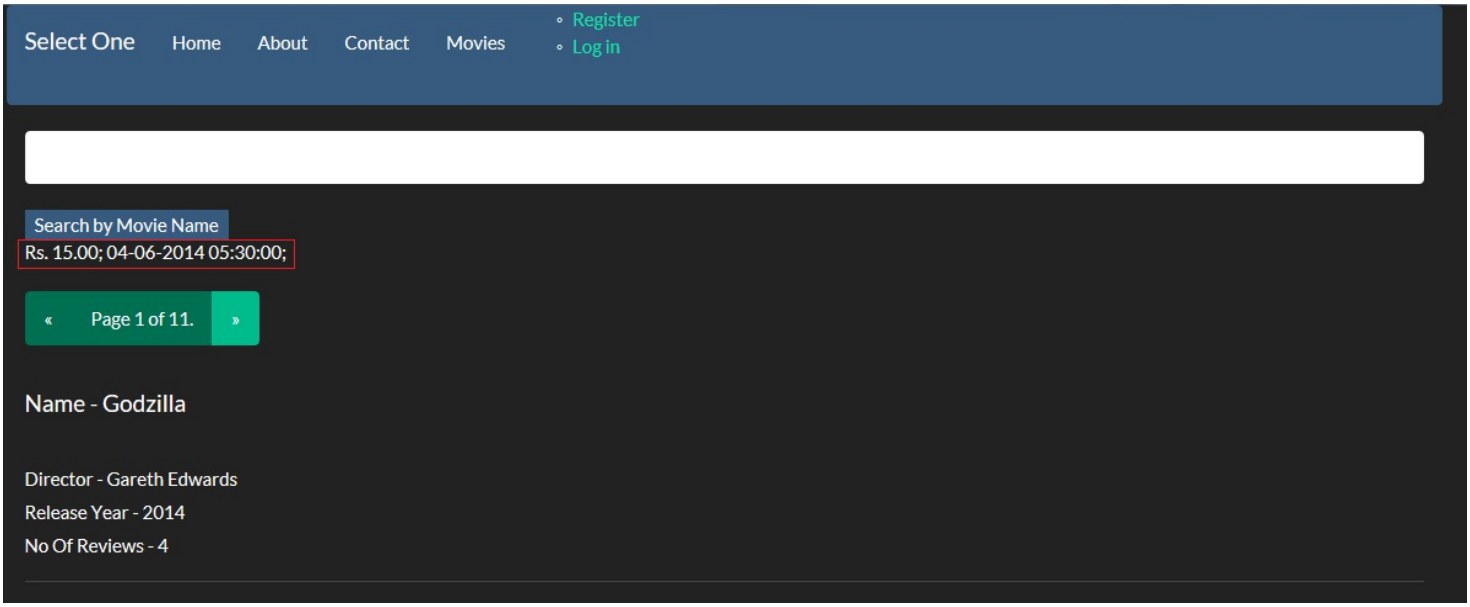

Internet Options
General
Security
Privacy
Content
Connections
Programs
Advanced
Home page
To create home page tabs, type each address on its own line.
about:blank
Use current
Use default
Use blank
Browsing history
Delete temporary files, history, cookies, saved passwords, and web form information.
Delete browsing history on exit
Delete...
Settings
Search
Change search defaults.
Settings
Tabs
Change how webpages are displayed in tabs.
Settings
Appearance
Colors
Languages
Fonts
Accessibility
OK
Cancel
Apply

Now, click on the language setting and add new language there and move the same up as shown below.



Now, with the above setting in place when I refresh the page then it will

produce the below output in Indian format.



Now, as far as localization is concerned, we are only half way done. We need to also see how we can change string literals like suppose if I have selected some language like "***French***" language, then in that case my string literals should also get translated into that locale. So, to localize the text I need to rely on the resource file or files ending with "***.resx***"

extension.

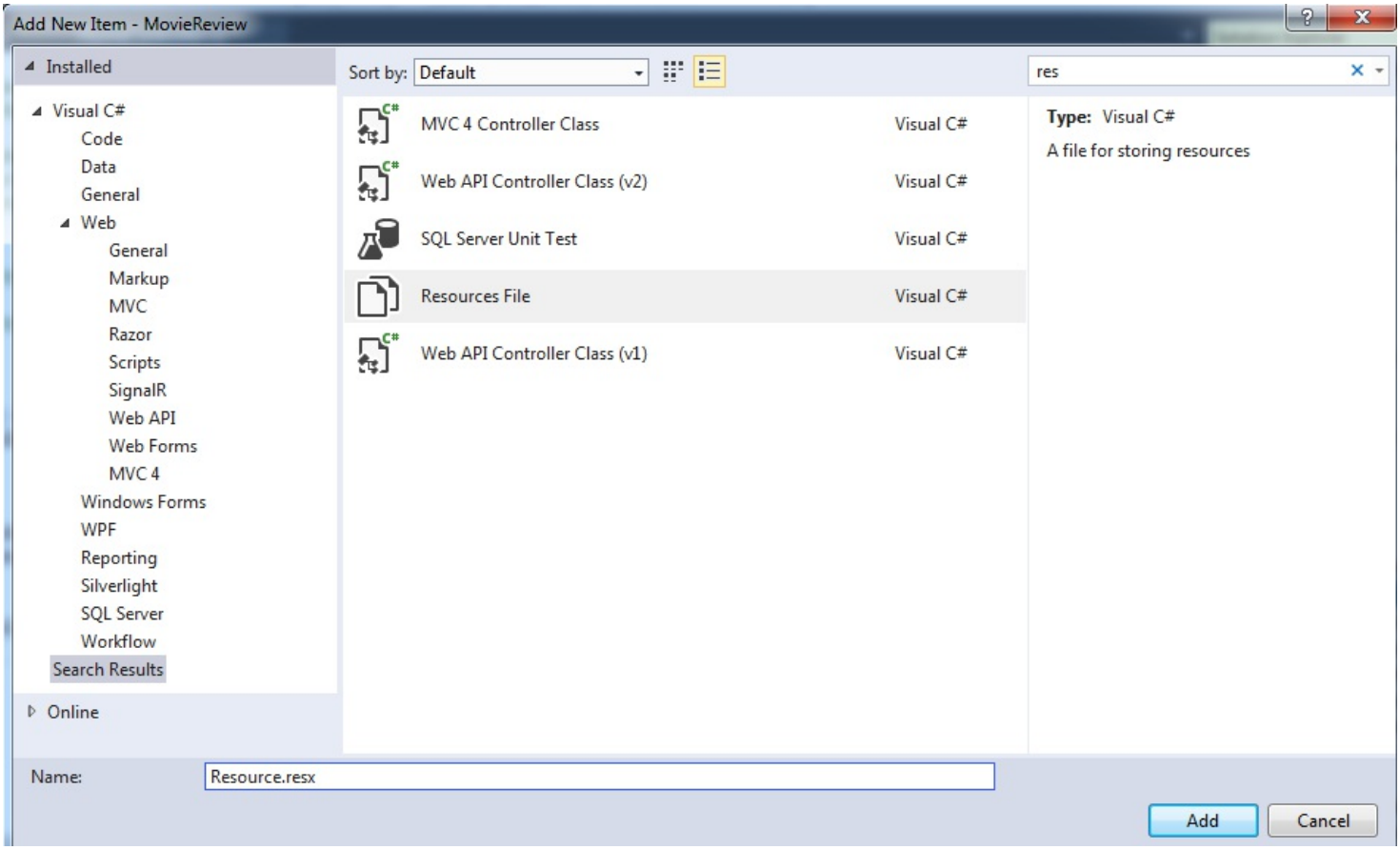
Resource Files:-

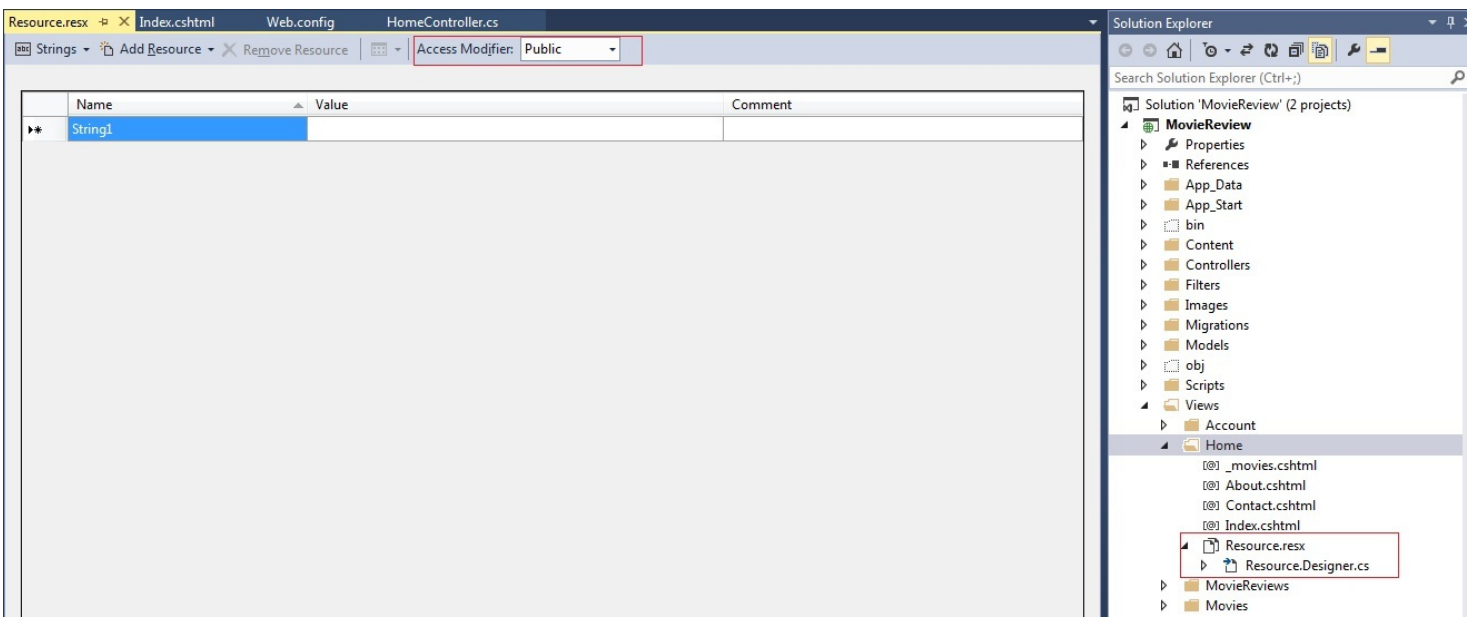
So, basically these resource files are xml files which stores localized strings for a specific culture. So, what I'll be doing here, I'll be creating two resource files; one is for our language "***English***" and another for other language "***French***". So, two files with the names "***Resource.resx***" and "***Resource.fr.resx***" are for English and French. So, this is the general naming convention for different resources. Now, in ASP.Net, we have resource manager which loads the associated resource file

for the current UI culture which is set. Now, these actions are backend actions, we generally don't have to interact with resource files directly, Visual Studio takes care of this by compiling the same for us in a strongly typed class. Now, let me explain the same with a small demo.

So, I am just going to output "***Hello***" on the screen in English version and French Version. Now for creating resource file, we can create the same inside the same project or in a different class library project for making clean and extendable. So, decision is completely on you how do you like to create. But, for demo purpose, I would like to create the same in the same project. So, here I'll be

adding the same in Home Views folder as shown below.

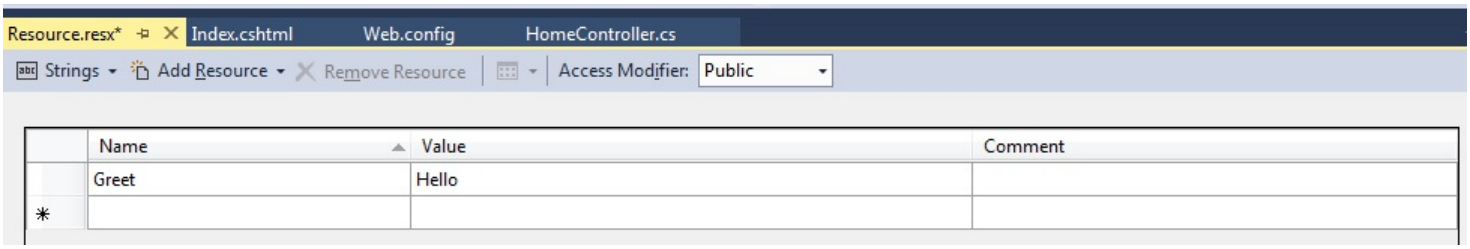




Now, when you open the file in Visual Studio, the very 1st thing which might want to change is the access modifier. By default it would be internal, which I would change it Public. So, this is one of the required things for compilation to happen smoothly. As we know that **Razor Views** are compiled in a different assembly basically a different file which is not in the scope of our project. So, in order to make these resources available in our razor views we have to make **Access Modifier** for these resources as **Public**.

Now, as you can see in the above screen shot, visual studio has also generated one class file "***Resource.Designer.cs***"

for us. We'll see the same in a bit. But, 1st set the strings which we want to reach from the view. So, in the below screen shot, I have set the value and then save the same. Once, I saved this file, you could see in the class file, it generated one property with the name "***Greet***" in the class file. So, now this is strongly typed and will be easily available from the view to load the specific resource.



```
        /// <summary>
        ///   Overrides the current thread's CurrentUICulture property for all
        ///   resource lookups using this strongly typed resource class.
        /// </summary>
        [global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]
        0 references
        public static global::System.Globalization.CultureInfo Culture {
            get {
                return resourceCulture;
            }
            set {
                resourceCulture = value;
            }
        }

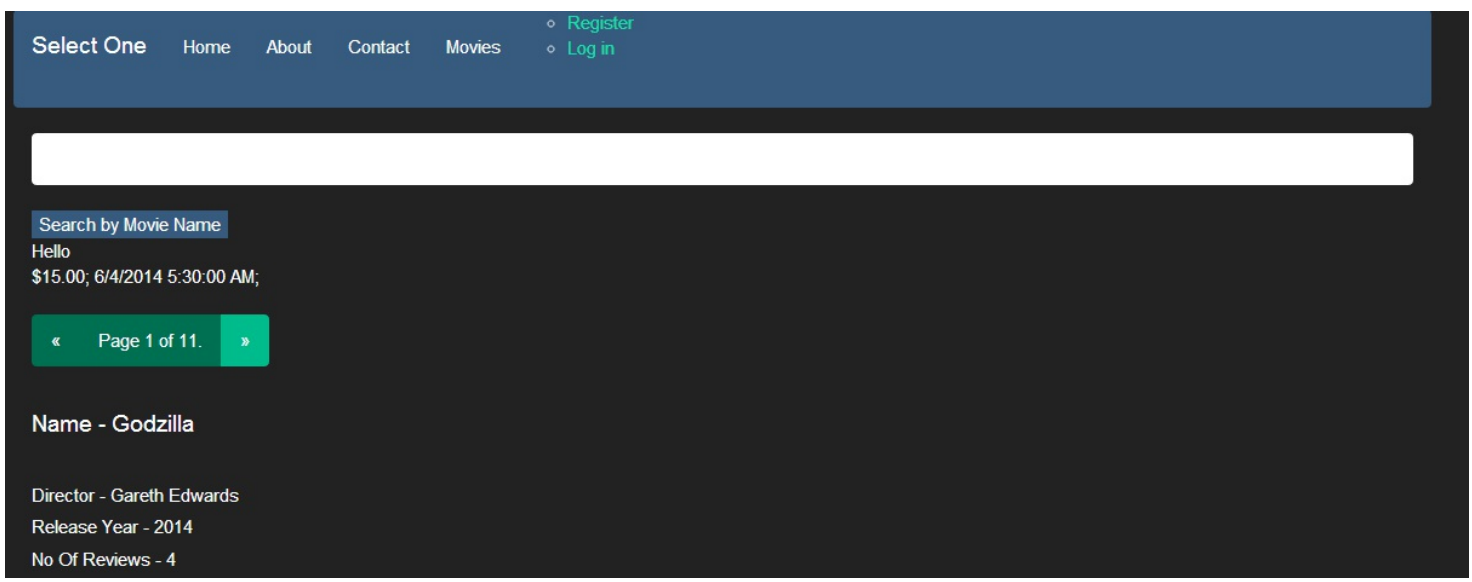
        /// <summary>
        ///   Looks up a localized string similar to Hello.
        /// </summary>
        0 references
        public static string Greet {
            get {
                return ResourceManager.GetString("Greet", resourceCulture);
            }
        }
    }
}
```

So, now what I do now, I would replace my hard coded text with the namespace, class name and then the property name which I am interested in as shown below.

```
@{

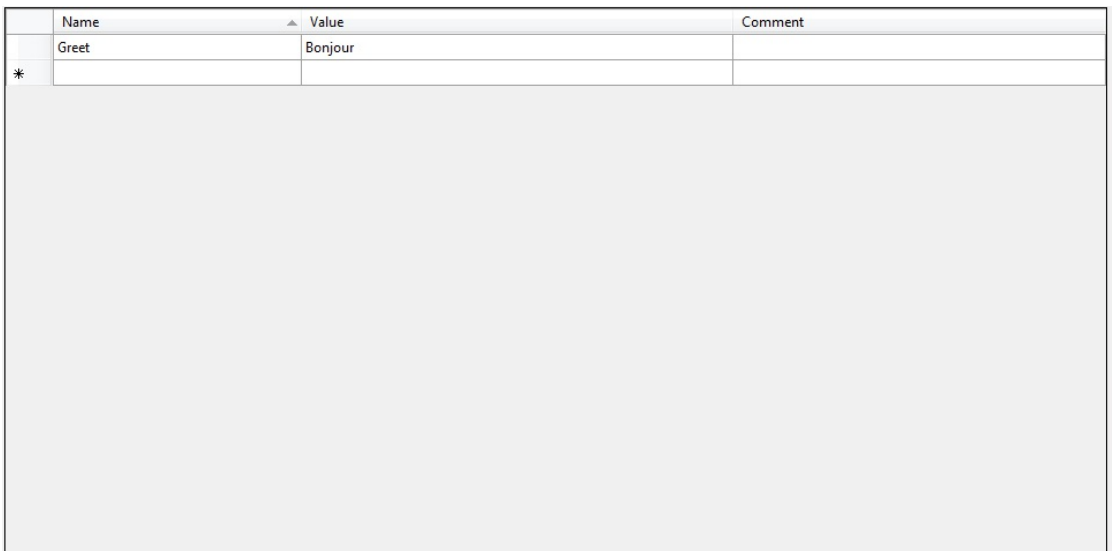
    var amnt = 15.0m;
    var localDate = new DateTime(2014, 6, 4);
}

<div>
    @MovieReview.Views.Home.Resource.Greet
    <br/>
    @amnt.ToString("c");
    @localDate.ToLocalTime();
</div>
```

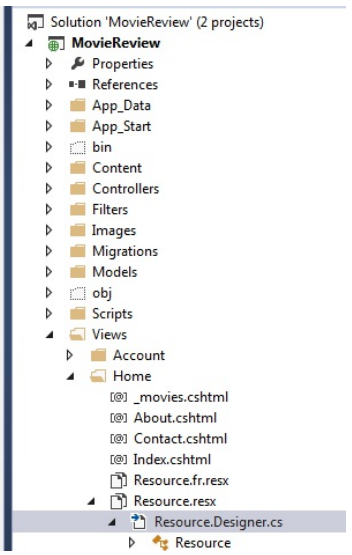


So, now if at this point if I run the

application, then it will display the message from the Resource file "***Hello***" irrespective of what language I have chosen. Because currently I have only one resource file and that will be treated as a base resource file. Now, if I want to customize the same for a different language, I need to create another resource file for a different language.

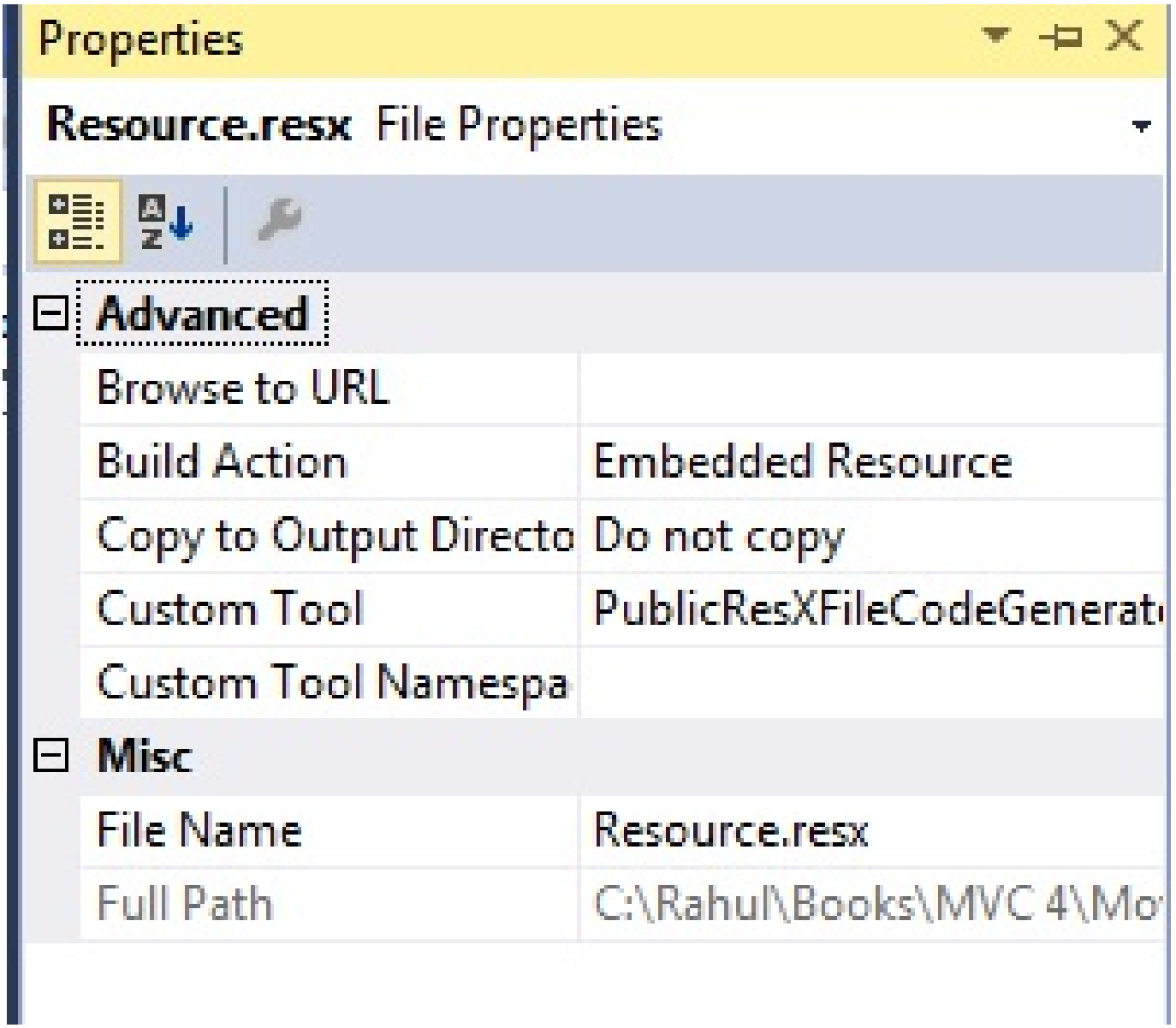


	Name	Value	Comment
	Greet	Bonjour	
*			



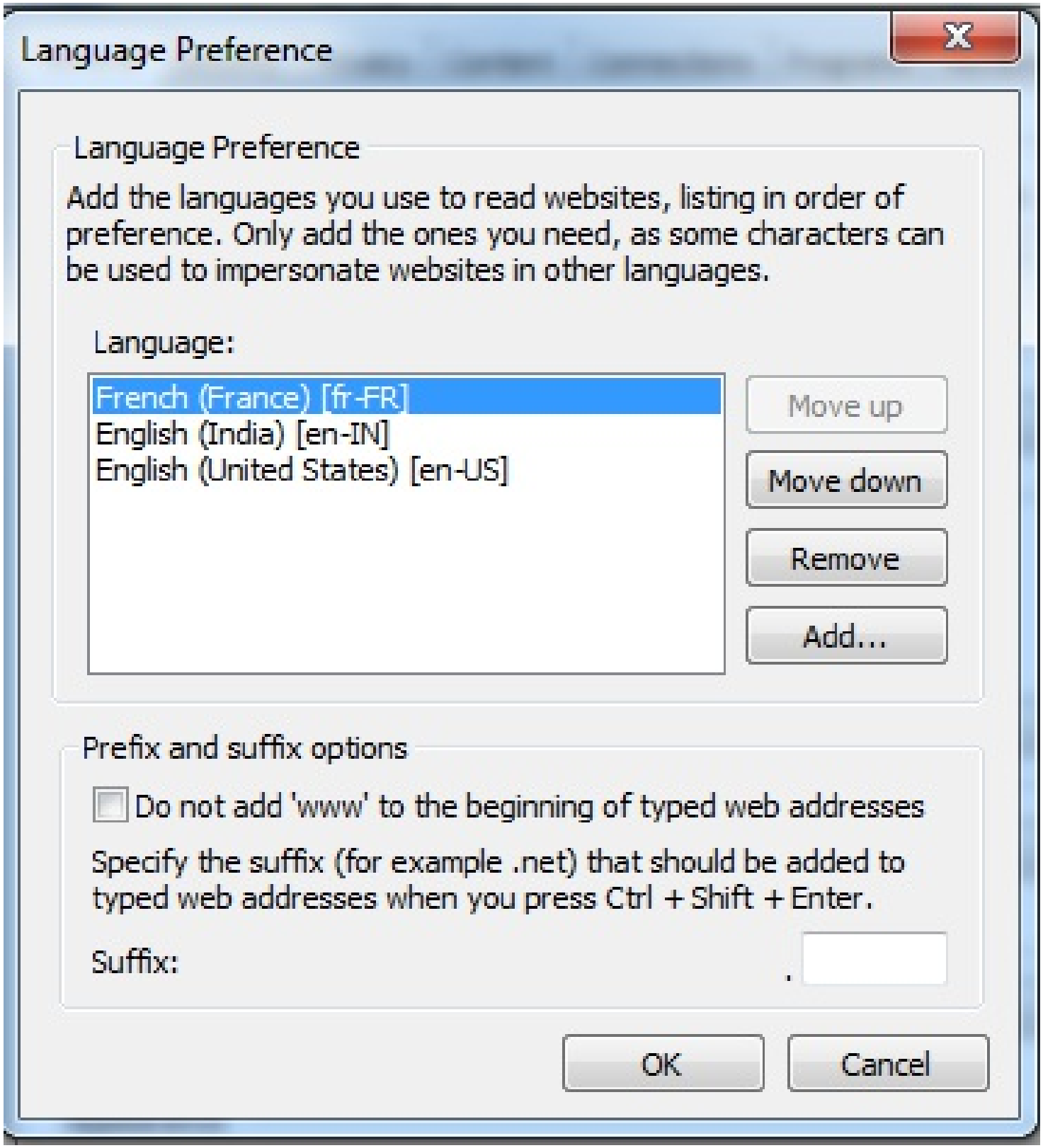
Now, all of these resource file will get embedded in the base resource file. So,

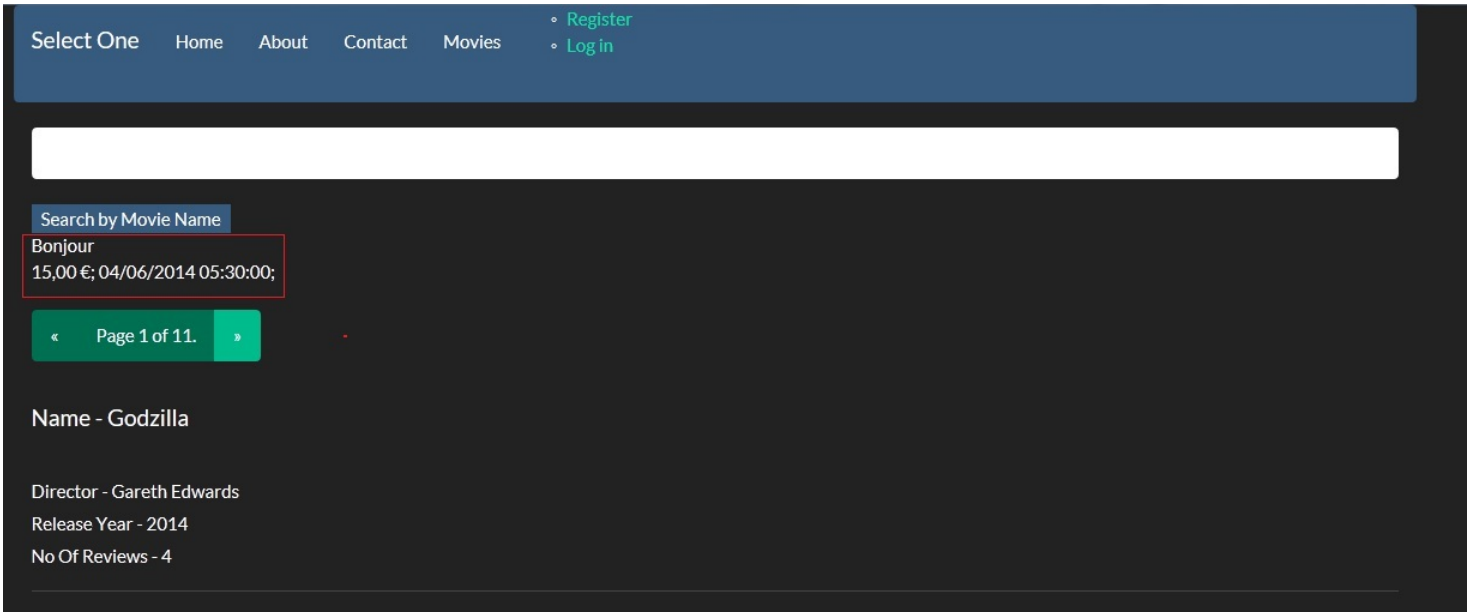
when you see the Resource file property, you will see that build action is "***Embedded Resource***" as shown below in the screen shot.



Now, let's run the application by setting

the 1st language as French from the IE language setting.





So, it worked. It gave me hello in French version; it also produced the euro symbol there. So, now these settings I can do from the controller action as well. Now, there could be chances that caching might interfere here, even if I set the language back to any other format. So, in order to fix this, we could pass "Accept-Language" from the index action itself as shown below.

```
[OutputCache(CacheProfile="Mild",Location=OutputCacheLocation.Server,VaryByHeader
= "Accept-Language")]
        public ActionResult Index(string SearchQuery = null,int page=1)
        {

            var model = _movieDb.Movies
                        .OrderByDescending(m => m.reviews.Count())
                    Where(m=>SearchQuery==null||m.MovieName.StartsWith(SearchQuery))
                        .Select(m => new MovieViewModel
                        {
                            Id = m.Id,
                            MovieName = m.MovieName,
                            DirectorName = m.DirectorName,
                            ReleaseYear = m.ReleaseYear,
                            NoOfReviews = m.reviews.Count()
                        }).ToPagedList(page,10);
            if (Request.IsAjaxRequest())
            {
                return PartialView("_movies", model);
            }
            return View(model);
        }
```

So, here I am telling the ASP.Net, clear the cache based on the header as well and that is based on the language. So, now if you build the application and try even within the caching time with separate language, this will load properly.

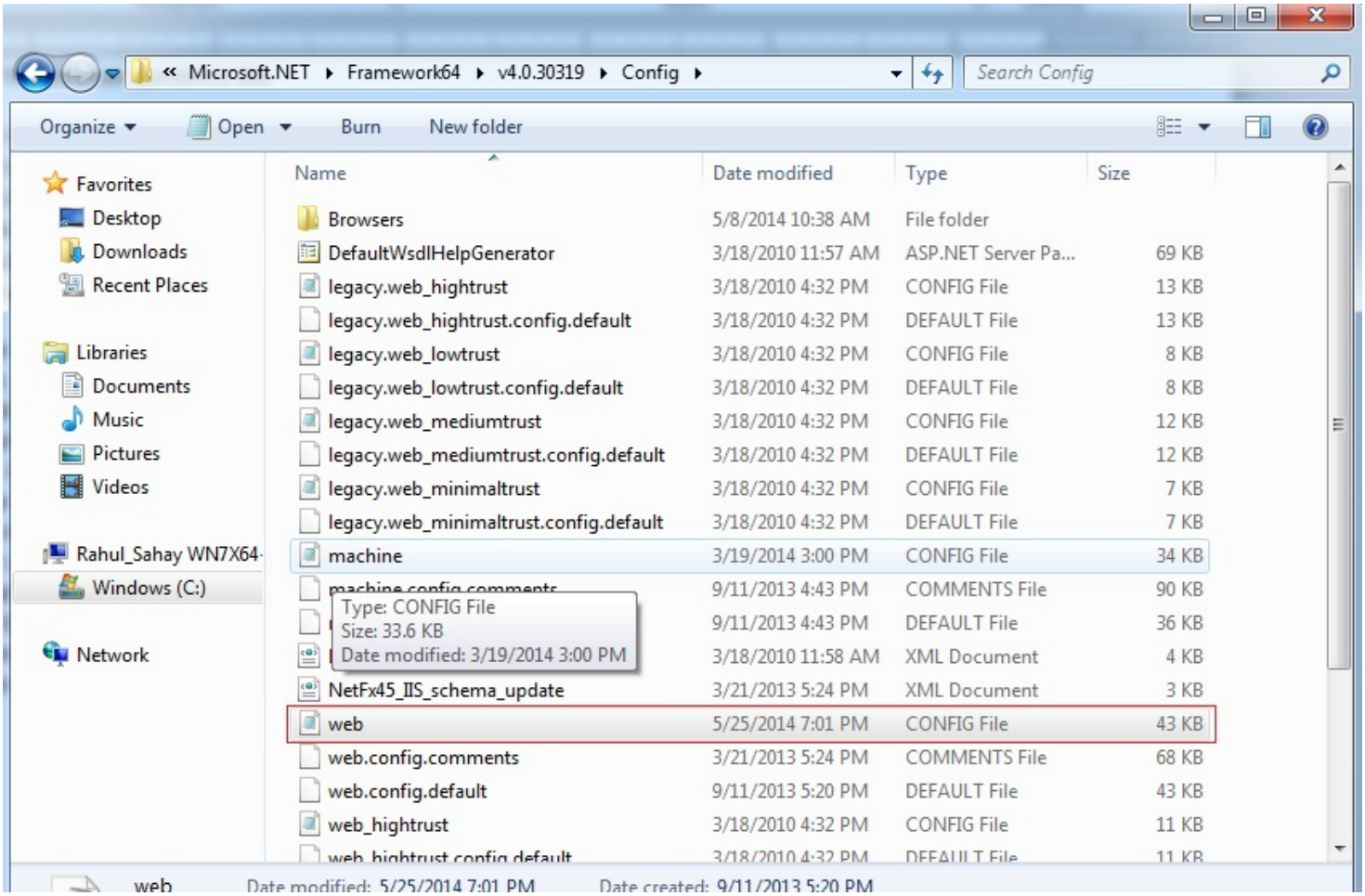
Diagnostics:-

Now, in this section we'll see some of the Diagnostics feature. Well this is the feature which gives you impression that how your ASP.Net MVC application is working means how healthy the application is. Potentially, we use this feature to track page traffics also page performance or in a nutshell application performance. So, basically Diagnostic answers few of very obvious questions like **When application started, When application crashed, When application aborted, Any exception encountered, Any authorization failure, And many more**.

So, to answer these kinds of questions we have number of options available. But, with ASP.NET, we have **Health Monitoring System** in place, which we can set from "***web.config***" file. Apart from this, there are also many third party tools to monitor the application, like "***Log4Net***" is popular open source software for logging, which you can use in your application. Another package, which I really like, is "***ELMAH***". ELMAH stands for "***Error Logging Modules And Handlers***". So, it basically captures events about your application and stores the same at variety of locations like xml files, text files or databases.

You can also share the results on social sites like **Twitter**. Let's see the same in action. So, the default ASP.Net configuration settings live in the "***web.config***" file. But, it's not the "***web.config***" file which sits in the root of your application. Now "***web.config***" file which applies configuration to every ASP.Net application which runs in your machine; it sits at the machine level. Now, in order to go this web.config file, we need to navigate to the following path "***C:\Windows\Microsoft.NET\Framew***
Now, it's pretty confusing with the Framework naming convention, although I am using **Framework 4.5**, but still it lives in the Framework 4 folder, with the

updates in place. Now, when you open, this folder, you can see web.config file as highlighted below in the screen shot.



Now, this is the file which is responsible to apply settings to all ASP.Net Applications. Now, when I open the same in Visual Studio, I can

search for the Setting "***HealthMonitoring***", and then it will give me the associated setting for the same as shown below.



So, as you can see that there are many settings associated to Health Monitoring like **Providers**, which determines where you want to save the logs; one option

could be in **Event Logs** which you could see from the settings or may be in **SQL Server** or may be in **WMI Events**. So, these are built in providers, but again you can add your custom providers here as well. Now, there is one more setting which describes the "***eventMappings***" of the application. So, it basically categorizes the same into different events as you can see below in the screen shot.



As you see in the screen shot, it is the rules section which is deciding where these events should go. Like it is saying that "***All Errors***" should go to "***EventLogProvider***". Now, again you can setup your own rules. Let me explain the same by creating one exception in the application. So, in the home controller under the index action, I'll intentionally

throw an exception as shown below.

```
[OutputCache(CacheProfile="Mild",Location=  OutputCacheLocation.Server,VaryByHeader
= "Accept-Language")]
        public ActionResult Index(string SearchQuery = null,int page=1)
        {
             throw new Exception("Log this Exception");
            var model = _movieDb.Movies
                        .OrderByDescending(m => m.reviews.Count())
                    .Where(m=>SearchQuery==null||m.MovieName.StartsWith(SearchQuery))
                        .Select(m => new MovieViewModel
                        {
                            Id = m.Id,
                            MovieName = m.MovieName,
                            DirectorName = m.DirectorName,
                            ReleaseYear = m.ReleaseYear,
                            NoOfReviews = m.reviews.Count()
                        }).ToPagedList(page,10);
            if (Request.IsAjaxRequest())
            {
                return PartialView("_movies", model);
            }
            return View(model);
        }
```

So, now, when I build and refresh the page, this should give me an exception as shown below.

Server Error in '/' Application.

Log this Exception

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

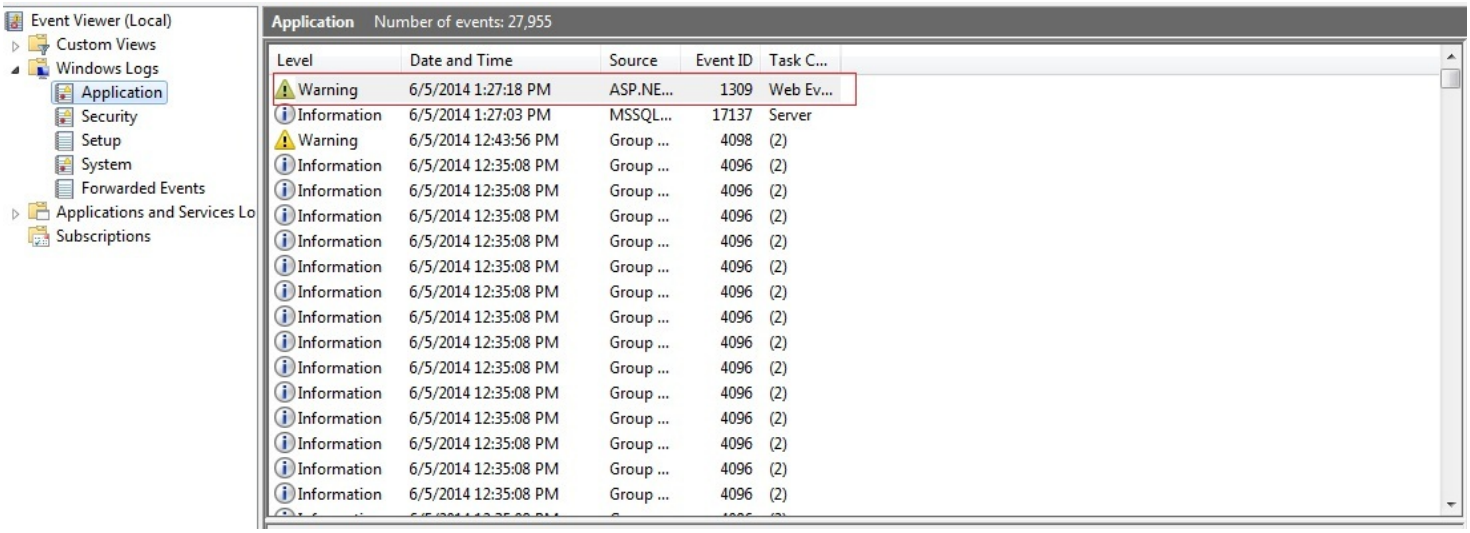
Exception Details: System.Exception: Log this Exception

Source Error:

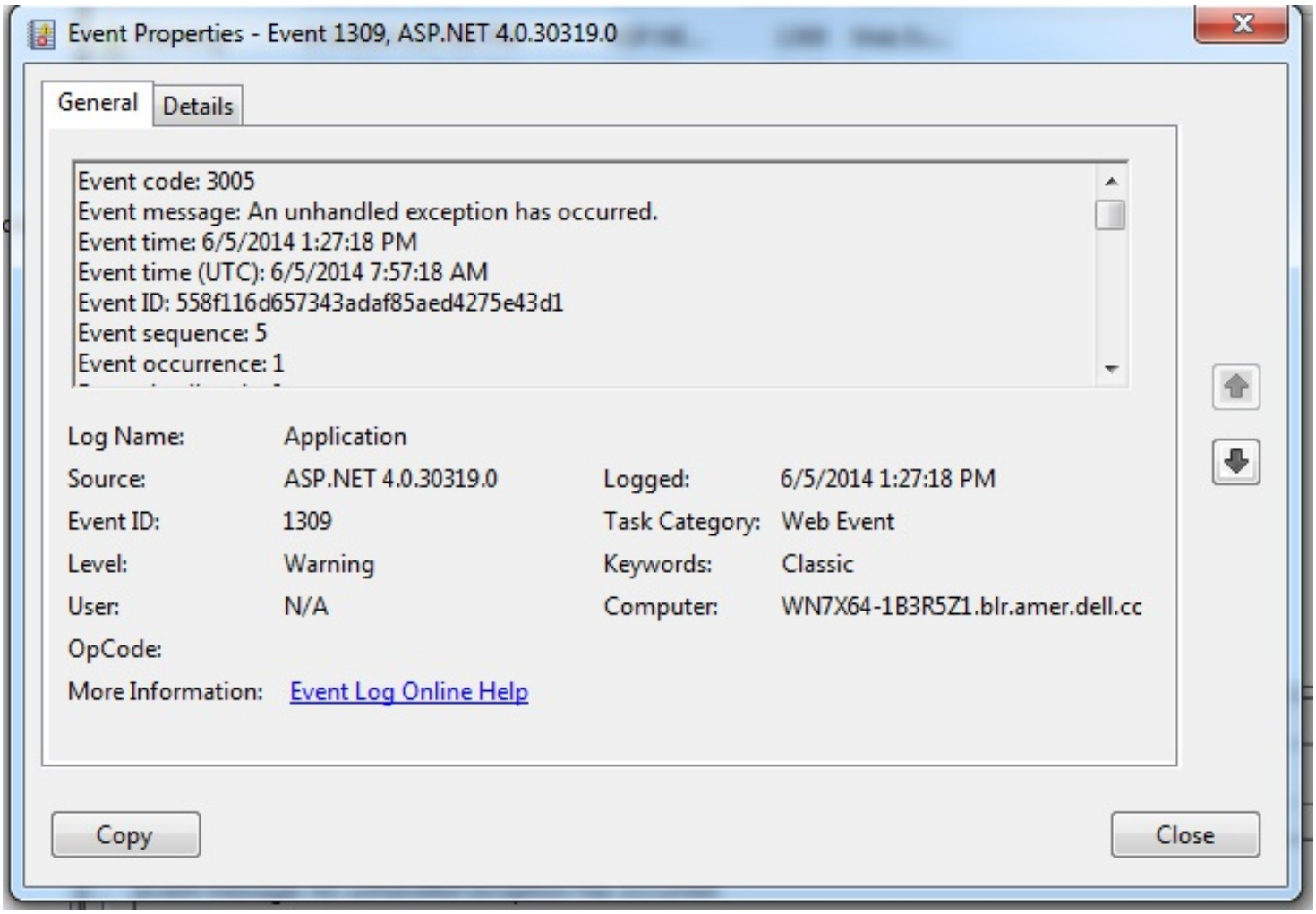
```
Line 17:         public ActionResult Index(string SearchQuery = null,int page=1)
Line 18:         {
Line 19:              throw new Exception("Log this Exception");
Line 20:             var model = _movieDb.Movies
Line 21:                         .OrderByDescending(m => m.reviews.Count())
```

Source File: c:\Rahul\Books\MVC 4\MovieReview\MovieReview\Controllers\HomeController.cs **Line:** 19

Now let me go ahead the see the event logs whether it recorded the problem or not. So, for doing the same I'll type "***eventvwr***" in **windows → Search Program and Files section**. Then, in that event viewer under the Windows "***Logs → Application***", you can find the error.



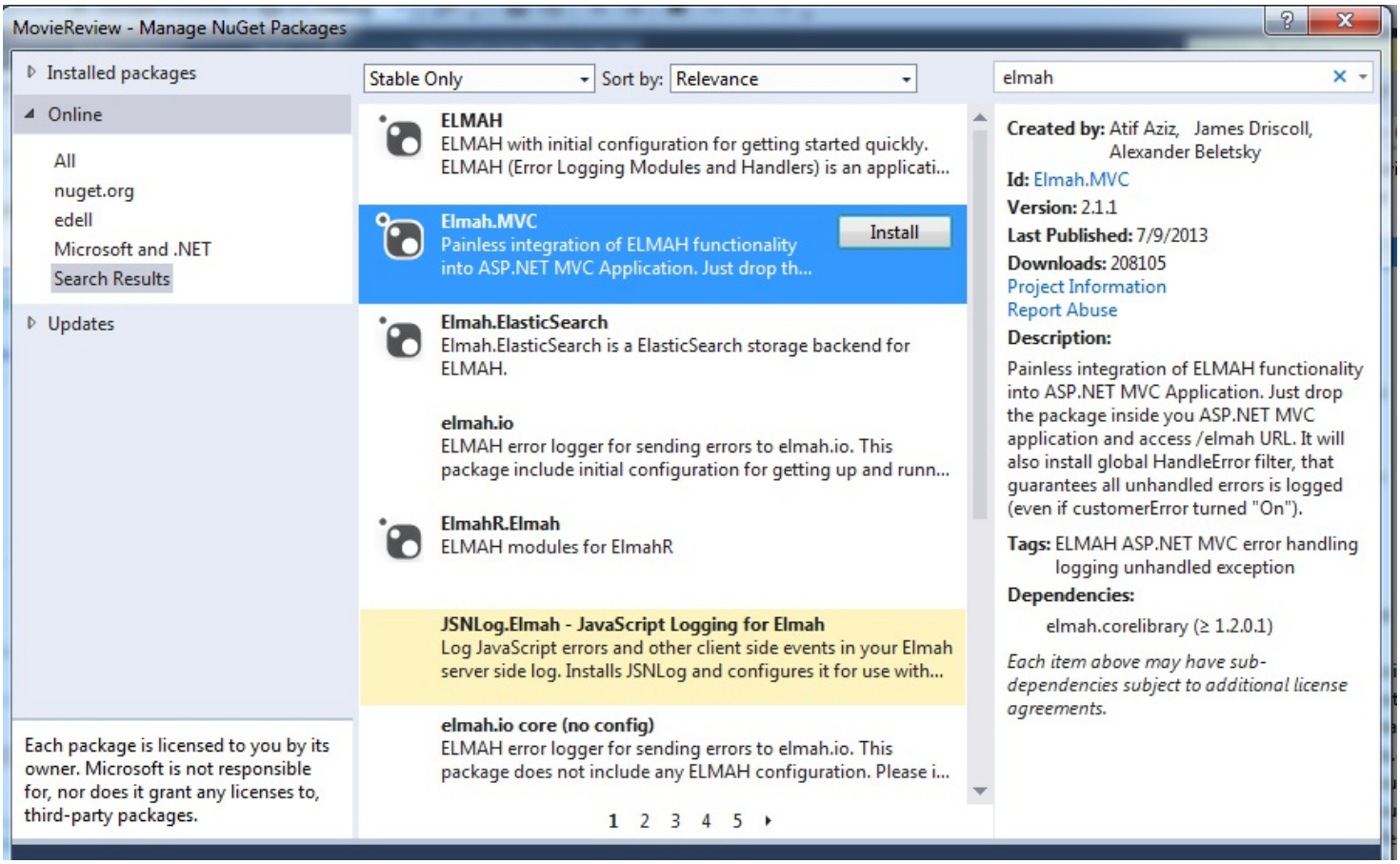
Now, when you double click the Highlighted Warning, it will list you the detail for the error as shown below in the screen shot.



So, this is fine. All these errors can be recorded in the events log. But sometimes it becomes quite tricky to

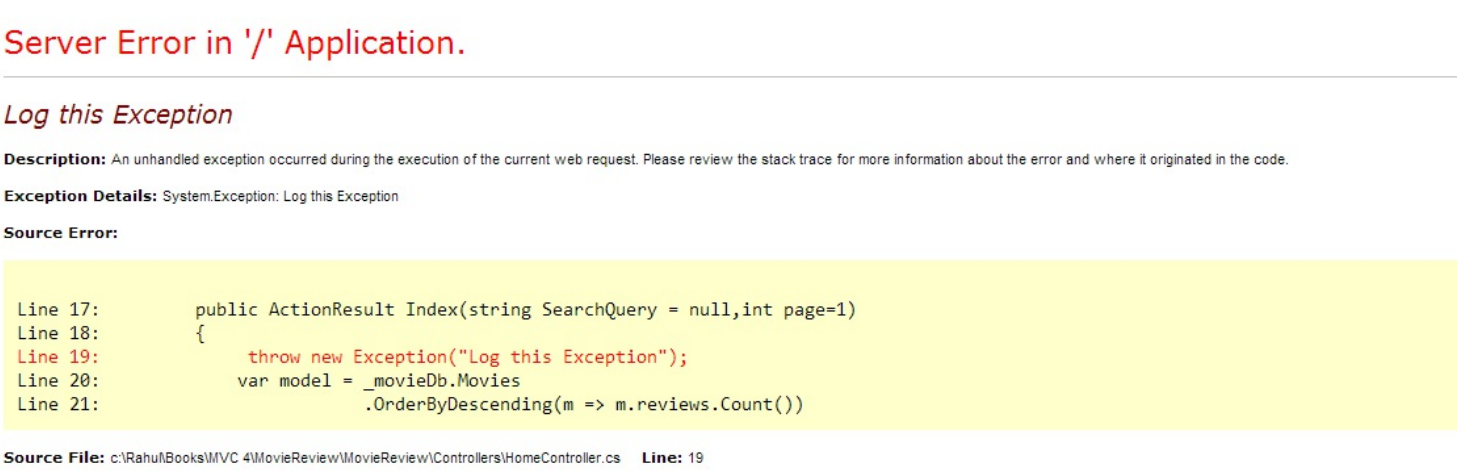
drill down the error from the event logs. At that moment we need a "***WEB UI***" kind of interface which will list all these events, exceptions and everything related to application. So, for these cases, we might need ELMAH enabled in our application. So, to install the same; the easiest way is "***nuget package manager***" as shown below in the screen shot.

ELMAH:-

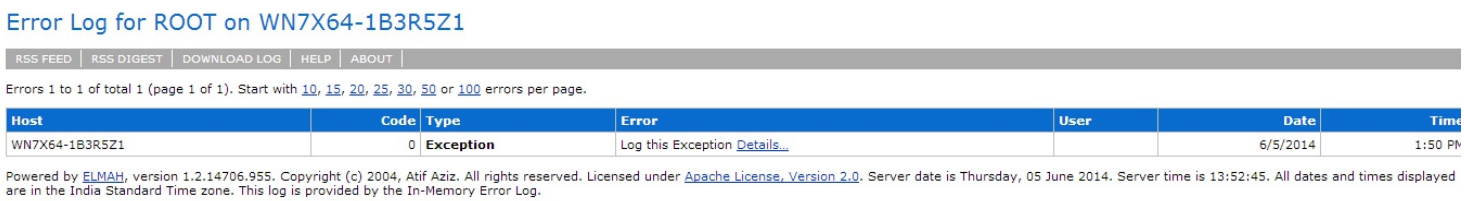


Now, once this is successfully installed, then I would just build the application once, just to make sure all required things are in place. Then I'll refresh the page again and it will show me the below error. One point of caution here, if this "***Elmah.MVC***" gives error like **no dll** is found after installation, then

install the root ELMAH as well 1st one from the list.

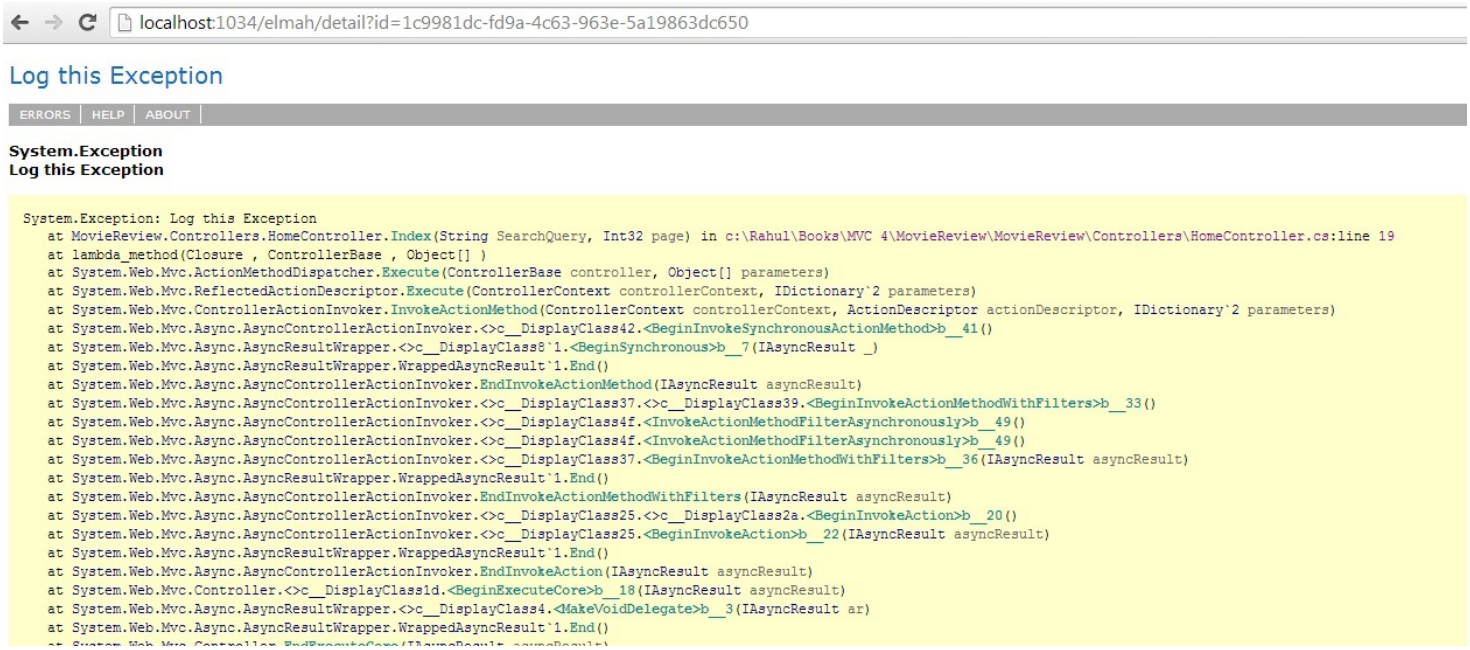


Then, you need to navigate to "***http://localhost:1034/elmah***", this will list the error in UI as shown below.



Now, when you click on the details link in the Error section, then it will list all

the details with the stack trace as shown below. So, this will help you to circumvent the problem.



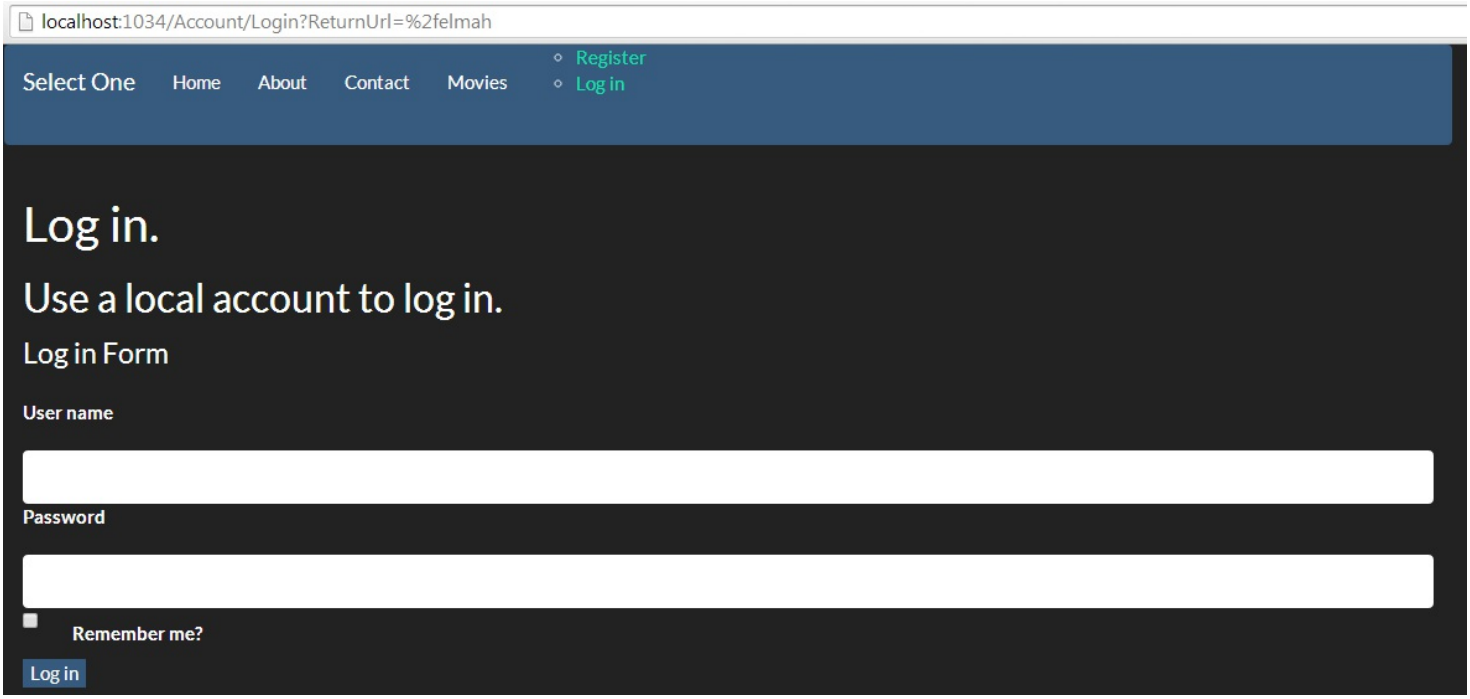
Now, you can also set the settings, like whenever any error happens, and then it should send you an email. So, all these settings can be done from the web.config file under ELMAH settings as shown below.

```
<configSections>
  <!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkID=2374
  <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFra
<sectionGroup name="elmah">
    <section name="security" requirePermission="false" type="Elmah.SecuritySectionHandler, Elmah" />
    <section name="errorLog" requirePermission="false" type="Elmah.ErrorLogSectionHandler, Elmah" />
    <section name="errorMail" requirePermission="false" type="Elmah.ErrorMailSectionHandler, Elmah" />
    <section name="errorFilter" requirePermission="false" type="Elmah.ErrorFilterSectionHandler, Elmah" />
  </sectionGroup></configSections>
```

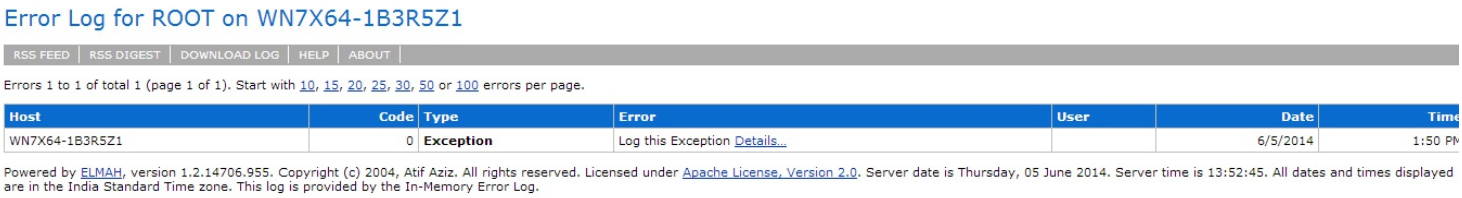
So, as you can see in the screen shot, there is variety of settings which you can administer from the web.config file in your application. Also, you could set that who is authorized to see these errors. Since I already have a setting for admin, so I can set that as well. I have also set **requires Authentication** to true, so that it will ask me to login before I can see the errors.

```
<appSettings>
  <add key="webpages:Version" value="2.0.0.0" />
  <add key="webpages:Enabled" value="false" />
  <add key="PreserveLoginUrl" value="true" />
  <add key="ClientValidationEnabled" value="true" />
  <add key="UnobtrusiveJavaScriptEnabled" value="true" />
  <add key="elmah.mvc.disableHandler" value="false" />
  <add key="elmah.mvc.disableHandleErrorFilter" value="false" />
  <add key="elmah.mvc.requiresAuthentication" value="true" />
  <add key="elmah.mvc.IgnoreDefaultRoute" value="false" />
  <add key="elmah.mvc.allowedRoles" value="Admin" />
  <add key="elmah.mvc.allowedUsers" value="*" />
  <add key="elmah.mvc.route" value="elmah" />
</appSettings>
```

So, with the above settings in place, when I refresh the application and navigate to **ELMAH**, it will bring me back to the login page and ask to login 1st.



Now, when I login, then it will show me the error page.



So, this way we can configure User interface for **Logging** and **Monitoring** at application level with ease.

Summary:-

So, in this section, we have seen how to use Caching. What different settings we can use in caching to boost the site performance. We have also seen how to use localization based on our culture and UICulture settings. In order to explain localization we have used different resource files with different locale values. Then, in the end we have seen how to use Diagnostics Settings. We have also seen Diagnostics at different level; at Event Viewer Level and at UI Level. We have implemented ELMAH to track the application at every instance.

Chapter 9: Unit Testing

WHAT DO you find in this CHAPTER?

⇒ Introduction

⇒ TDD

⇒ Home Controller Tests

⇒ Web Tests

⇒ Load Tests

⇒ Code Coverage

⇒ Summary

Introduction:-

Hi Friends, so, we have now done with the development in this case study. Now, it's time to write Test cases. So, in this section, we'll see that how to use **Test Driven Development** to drive the design of a feature. We'll also discuss

“**TDD (Test Driven Development)**” cycle. TDD Cycle comprise of three basic things. They are

Red: - First you write failing test, which gives you red result.

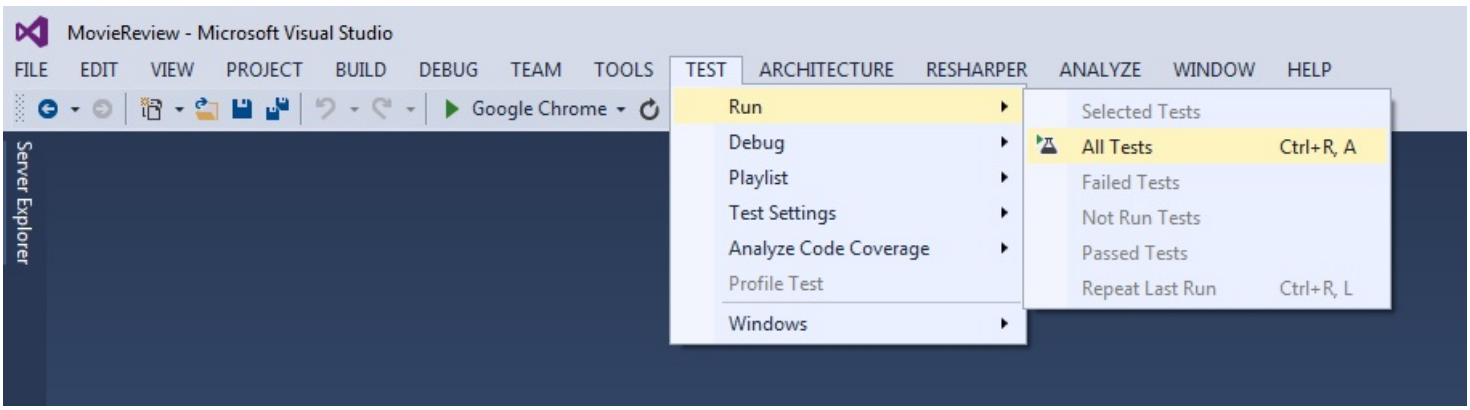
Green: - Second you write fulfilling the request; hence test passes and gives green result.

Refactor: - Then we do refactoring to improve the design, especially reusability in this case.

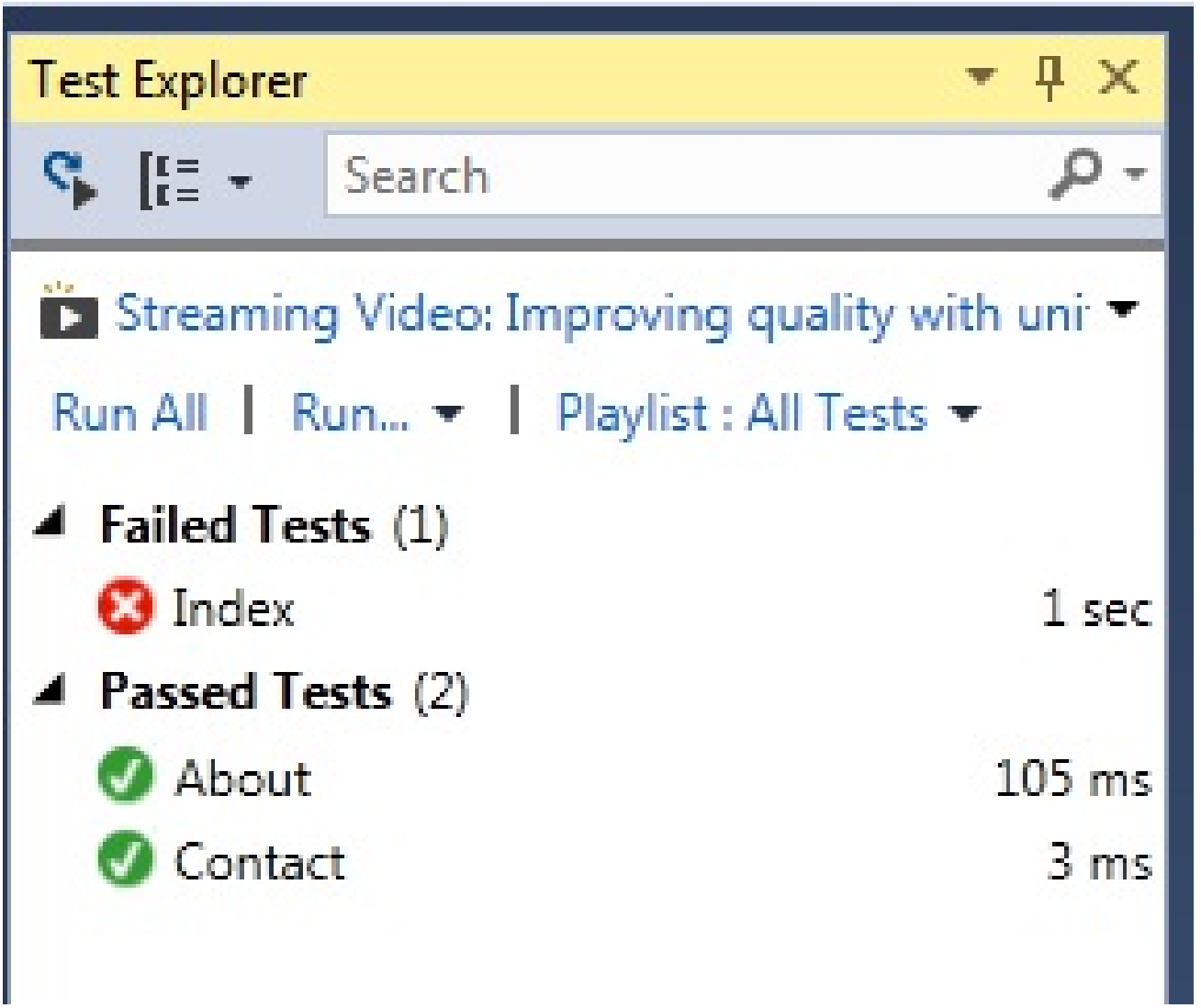
So, let's begin the chapter with TDD design.

TDD:-

TDD means Test Driven Development. So, as I started discussion on MVC, I emphasized on the topic, that whole idea of writing any MVC application is its TDD design. It enables user to write Test cases separately for any scenario. So, for this reason, we have created one test project along with our web project. So, we can go ahead and run the test cases from the menu as shown below in the screen shot.

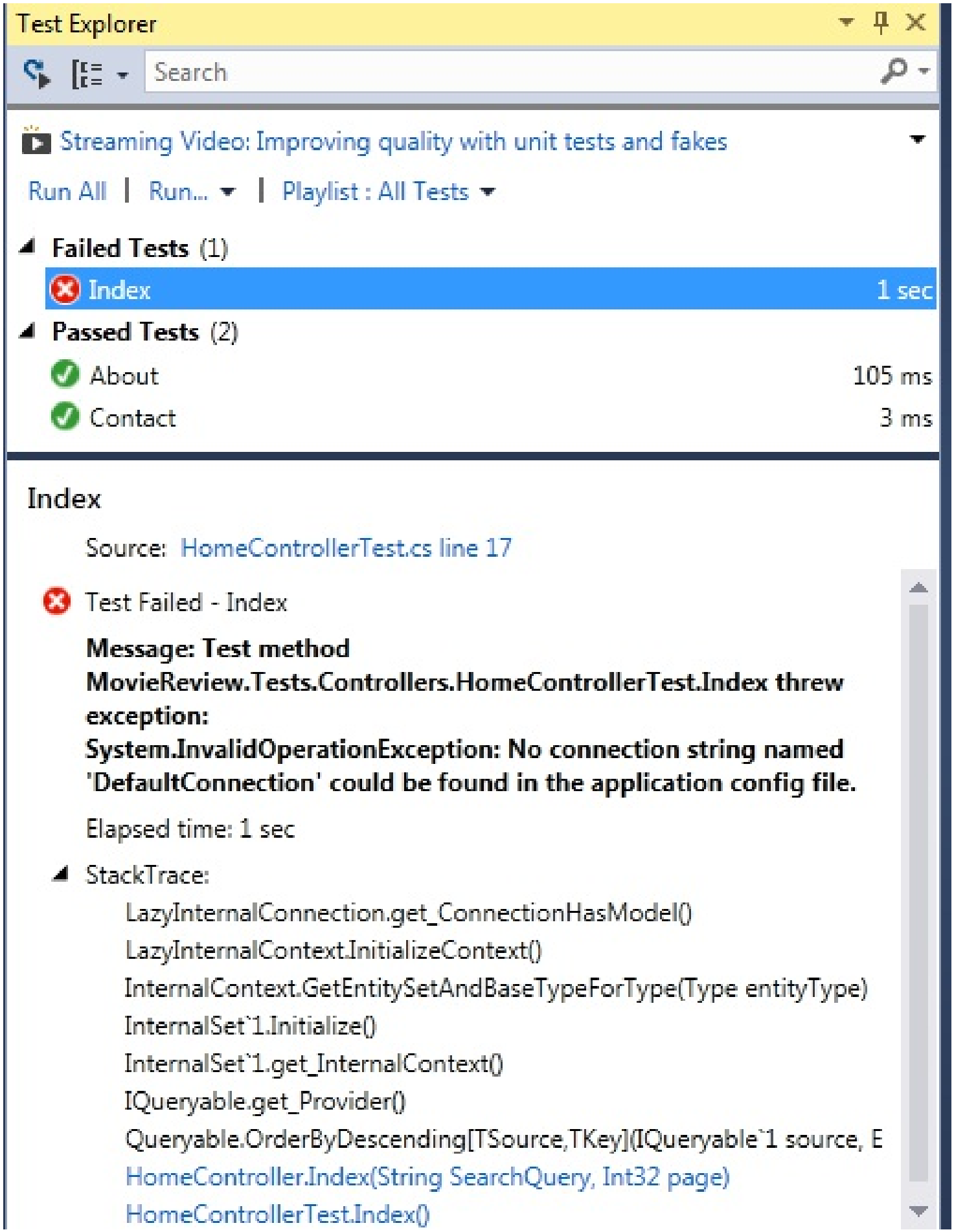


Now, when I ran the test, it produced me the below result.



So, as you can see in the above screen shot, two test cases passed and one failed. The other two test cases passed because we haven't changed anything in

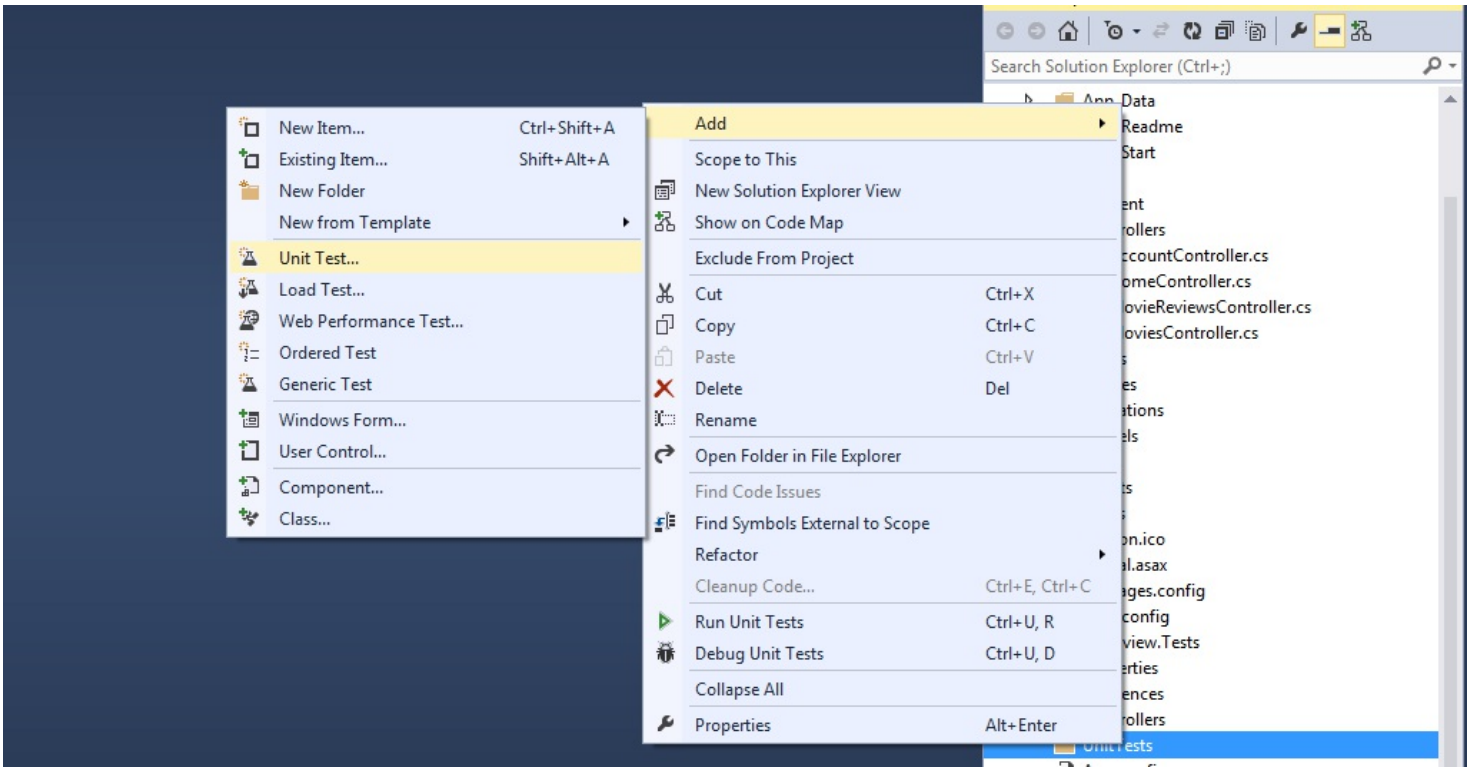
these two actions, these are as it is as presented by Visual Studio. But, Index action we have changed completely. So, in order to fix the same we need to modify this Test case. Now, in order to see the exception, you need to click on the index action here in the Test Explorer. So, when you click on it will produce the below result.


Test Explorer
Search
Streaming Video: Improving quality with unit tests and fakes
Run All | Run... | Playlist : All Tests
Failed Tests (1)
Index 1 sec
Passed Tests (2)
About 105 ms
Contact 3 ms
Index
Source: HomeControllerTest.cs line 17
Test Failed - Index
Message: Test method MovieReview.Tests.Controllers.HomeControllerTest.Index threw exception:
System.InvalidOperationException: No connection string named 'DefaultConnection' could be found in the application config file.
Elapsed time: 1 sec
StackTrace:
LazyInternalConnection.get_ConnectionHasModel()
LazyInternalContext.InitializeContext()
InternalContext.GetEntitySetAndBaseTypeForType(Type entityType)
InternalSet`1.Initialize()
InternalSet`1.get_InternalContext()
IQueryable.get_Provider()
Queryable.OrderByDescending[TSource,TKey](IQueryable`1 source, E
HomeController.Index(String SearchQuery, Int32 page)
HomeControllerTest.Index()

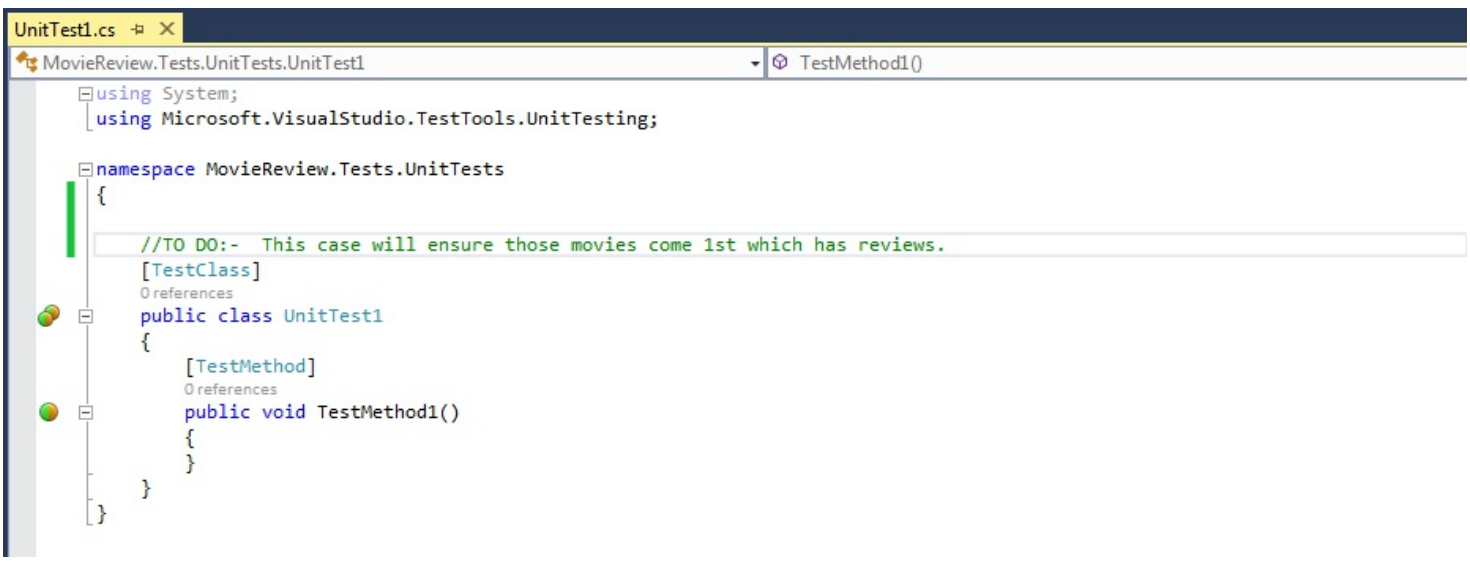
So, exception says Home controller tried to reach out the database, but couldn't find the connection string which it was looking for. And this test is invoking the home controller index action. But, next question you ask that how it is possible, because we have connection string defined for the web application and web application just works fine. So, how come exception is coming while running test. To answer this question, test is failing because we haven't configured this connection string for test.

But, now am going to halt a bit and we'll take this problem some time later. For now, just focus on the topic TDD. I urge all the readers to make this TDD

approach a habit while developing any web application as this is very beneficial. Writing Test cases or good Test cases covering your entire code ensures that your code will not break in those situations which you have tested with the test cases. Hence, it ensures the code which you are writing is quality code. Now, I have created one "***UnitTests***" folder in my unit test project. Now, in this I'll add my 1st unit test as shown below in the screen shot.



Now, in here my test, I'll be writing the test in such a way, that it should produce the movie which has the highest no of reviews 1st.



As you can see in the above screen shot Visual studio has already given me the Test class with all the test attributes in place. Now, I'll do a bit modification here

```
using System;
using System.Collections.Generic;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using MovieReview.Models;

namespace MovieReview.Tests.UnitTests
{

    //TO DO:-  This case will ensure those movies come 1st which has reviews.
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void TestMethod1()
        {
// Organise

            var data = new Movie();
            data.reviews = new List<MovieReviews>();
            data.reviews.Add(new MovieReviews(){ReviewerRating = 5});

            //Act

            var rater = new RateMovie(data);
            var result = rater.EvaluateRating(5);

            //Assert

            Assert.AreEqual(5,result.Rating);
        }
    }
}
```

Now, as you can see in the above method. 1st have created an instance of

Movie class. Now, when that instance is created then I created a list of **Reviews** for that **Movie** and then I added review for that movie. For now, I just created couple of reviews with rating 5. Now, I need API which will evaluate the movie rating. So, I'll write one more class to do the same. Eventually, these APIs basically sit in Web Applications, Business logic, but for now this is fine. So, below as you can see that I have created one class as "***RateMovie***" and "***RatingResult***".

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MovieReview.Models;

namespace MovieReview.Tests.UnitTests
{
    public class RateMovie
    {
        private Movie _data;

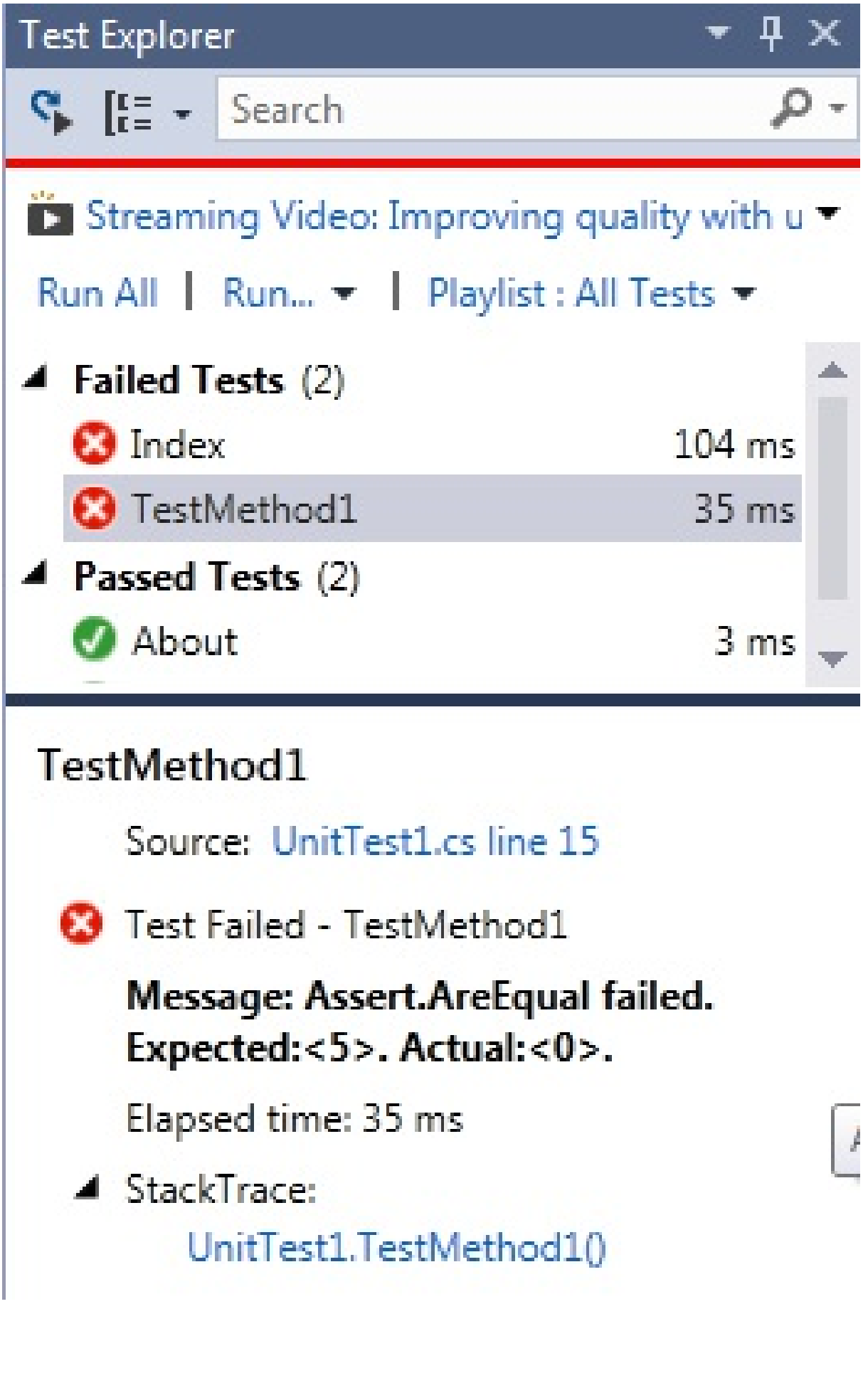
        public RateMovie(Movie data)
        {
            _data = data;
        }

       public RatingResult EvaluateRating(int rating)
        {
            return new RatingResult();
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace MovieReview.Tests.UnitTests
{
    public class RatingResult
     {
        public int Rating { get; set; }
     }
}
```

So, now I can go ahead and run the test, then it will produce me the below result as shown below.


Test Explorer
Search
Streaming Video: Improving quality with u
Run All | Run... | Playlist : All Tests
Failed Tests (2)
Index 104 ms
TestMethod1 35 ms
Passed Tests (2)
About 3 ms
TestMethod1
Source: UnitTest1.cs line 15
Test Failed - TestMethod1
Message: Assert.AreEqual failed.
Expected:<5>. Actual:<0>.
Elapsed time: 35 ms
StackTrace:
UnitTest1.TestMethod1()

So, it said while doing Assert, the expected result is 5 but, it is getting 0. So, now I would go ahead and assign some dummy value as 5, so that test case can pass as shown below. Because, currently in my business design I don't have anything or any logic which actually calculates rating, it just returns rating as user enters it. But, let's suppose any scenario where in you are validating any actual business scenario, then in that case, you will be calling that business class to evaluate your inputs and give the result, then with that item, you can go ahead do your verification or assertion. But, this is how TDD works. So, now I have modified my "***RateMovie***" class, now I am explicitly assigning value in

there as shown below.

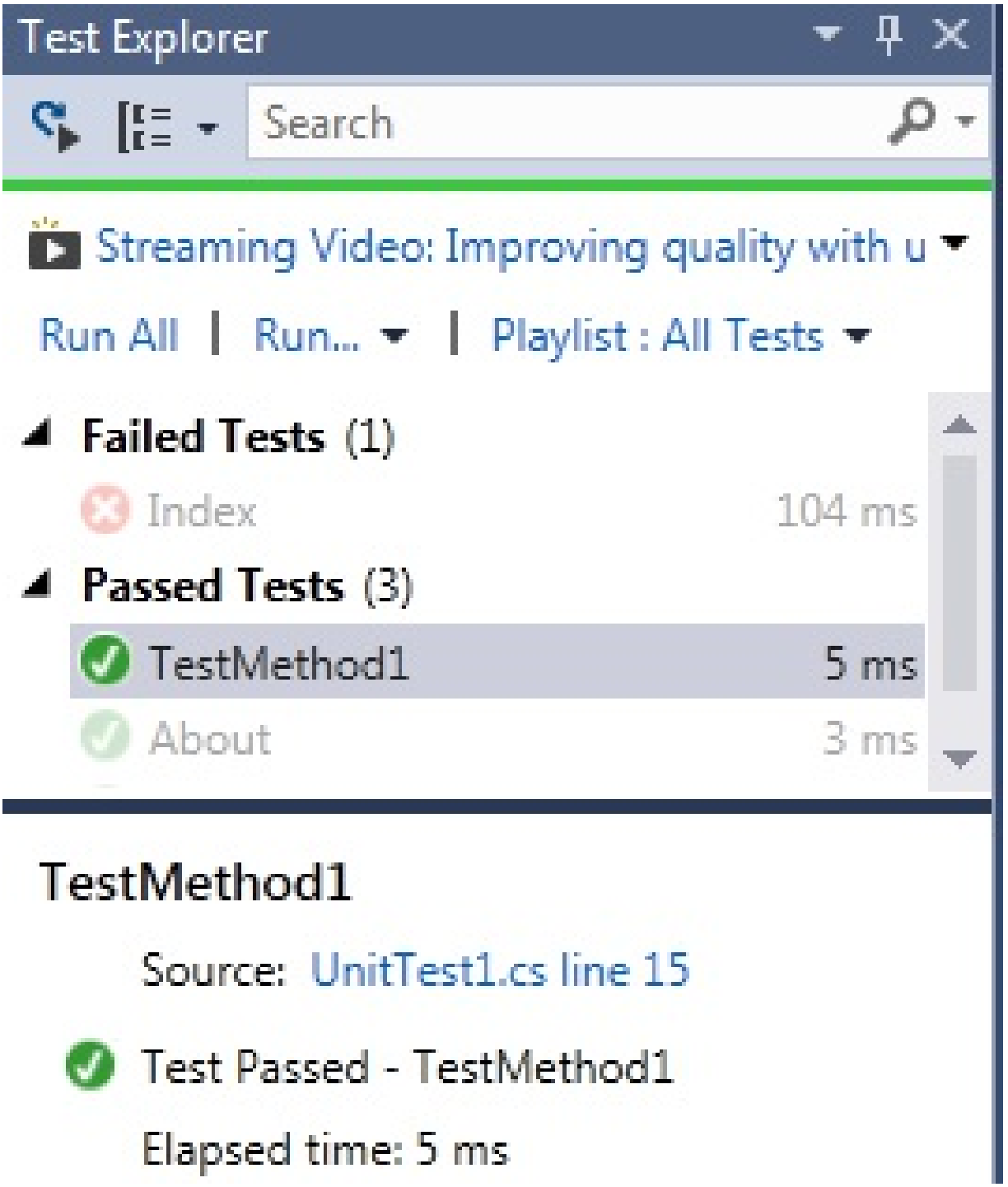
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MovieReview.Models;

namespace MovieReview.Tests.UnitTests
{
    public class RateMovie
    {
        private Movie _data;
```

```
        public RateMovie(Movie data)
        {
            _data = data;
        }

        public RatingResult EvaluateRating(int rating)
        {
            var result = new RatingResult();
            result.Rating = 5;
            return result;
        }
    }
}
```

Now, when I run the test, then it should pass now. So, in the below screen shot it is reflecting "***TestMethod1***" as passed result.



Now, by seeing this approach you will say that you are just hard coding the values to make sure that test are passed.

But, the beauty of TDD is this, you start small, even the simplest smallest possible thing you implement and then start adding conditions to it. So, the idea behind writing Test cases is let's suppose in future you are adding any new functionality to the application, and then in that case also your all test cases should pass. So, this basically ensures that your change is not breaking any existing thing. Now, once you have Passing Test in there then you can go ahead and start refactoring the code. In this case, I'll just rename few variables. So, below is the modified code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MovieReview.Models;

namespace MovieReview.Tests.UnitTests
{
    public class RateMovie
    {
        private Movie _movie;

        public RateMovie(Movie movie)
        {
            _movie = movie;
        }

       public RatingResult EvaluateRating(int numberOfReviews)
       {
           var result = new RatingResult();
           result.Rating = 5;
           return result;
       }
    }
}
```

So, as you can see that now it looks meaningful. So, another technique

writing any test class is that **Method Names** or **Test Classes** or **Variables** should be descriptive enough so that any developer can understand the whole picture by just looking the same.

Now, I have added 2nd test in the unit test module. So, this one is responsible for passing two reviews as shown below.

```
[TestMethod]
        public void Evaluate_Result_For_Multiple_Reviews()
        {
             // Organise

            var data = new Movie();
            data.reviews = new List<MovieReviews>();
            data.reviews.Add(new MovieReviews() { ReviewerRating = 5 });
            data.reviews.Add(new MovieReviews() { ReviewerRating = 3 });
            //Act
```

```
    var rater = new RateMovie(data);
    var result = rater.EvaluateRating(5);

    //Assert

    Assert.AreEqual(4, result.Rating);

}
```

Now, I have also given these test methods a descriptive name as you can see above. And, I have also modified the implementation of "***EvaluateRating***". So, it simply returns the average of reviews getting passed in using LINQ Query.

```
public RatingResult EvaluateRating(int numberOfReviews)
    {
        var result = new RatingResult();
        result.Rating = (int) _movie.reviews.Average(m => m.ReviewerRating);
        return result;
    }
```

Now, both the tests are passing. So, at this stage I can go ahead and start doing refactoring by cleaning up some code and making the same API based. So, as you can see below I have added one helper method just to make sure that it can take ratings and pass to multiple methods. It doesn't matter, whether you are passing single review or multiple reviews.

```
private Movie SetupMovieAndReviews(params int[] ratings)
{
     var movie = new Movie();
     movie.reviews = ratings.Select(m => new MovieReviews() {ReviewerRating = m})
     .ToList();
     return movie;
}
```

```
    //TO DO:-  This case will ensure those movies come 1st which has reviews.
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void Evaluate_Result_For_One_Review()
        {
            // Organise

            var data = SetupMovieAndReviews(ratings: new[] {5});

            //Act

            var rater = new RateMovie(data);
            var result = rater.EvaluateRating(5);

            //Assert

            Assert.AreEqual(5,result.Rating);
        }

        [TestMethod]
        public void Evaluate_Result_For_Multiple_Reviews()
        {
            // Organise

            var data = SetupMovieAndReviews(ratings: new[] {3, 5});

            //Act

            var rater = new RateMovie(data);
            var result = rater.EvaluateRating(5);

            //Assert

            Assert.AreEqual(4, result.Rating);
        }

        private Movie SetupMovieAndReviews(params int[] ratings)
        {
            var movie = new Movie();
            movie.reviews = ratings.Select(m => new MovieReviews() {ReviewerRating
= m})
                .ToList();
            return movie;
```

```
        }
    }
}
```

So, as you can see that I have used the helper method just to make sure that I can pass in multiple ratings or single rating with that method call itself. Hence, I saved couple of lines of extra code execution with refactoring technique.

Home Controller Tests:-

Now, the next test thing which I would like to do here is I need to fix my Index

Action test which is failing now. So, basically whenever any infrastructure is getting involved while writing any Unit Test Case, you need to isolate them; here in this case it is database connection. So, here we can take two approaches, we write the unit test in such a way that it hit the controller and go to the database. But, the problem with this approach is like unit test will run a bit slower because it will follow the complete execution path. And Unit Test means it should run really fast.

But, there is another problem also associated with this. Since, you are writing the test, so you want the controller to react predictably which

means you need to have valid data inside the database and database setup is proper. So, in this case unit test case will always be dependent on the database and connecting to the database would make test to run slow.

Now, the 2nd option is **Mocking**. Mocking here means whenever I am running any test I am not required to connect to database, I'll be using some sort of in memory data to validate the same. And, while running the application, it can refer the database and while testing it will test with mocked data. So, now the thing here is I need to remove the database. And removing the database needs some sort of interface

implementation. So, the interface will go in the file "***MovieDb.cs***" as shown below

```
public interface IMovieDb : IDisposable
    {
        IQueryable<T> Query<T>() where T : class;
    }
```

Now, this will represent all the operations which we want to do against the real database. So, here I just have the Query of "***T***" method. Now, here I want to query different entities like "***movies, movieReviews***". So, now my "***MovieDb***" class which we have been using so far will implement "***IMovieDb***" as shown below.

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Security.AccessControl;
using System.Web;

        }
    }
}
```

So, up there in the modified implementation, you can see that "***MovieDb***" is now using "***IMovieDb***". Here, it implements the query method explicitly means you can only come to this query method through "***IMovieDb***" reference. Now, the next thing which I

would like to change here is in the "***Home Controller***", where in I need to reference the "***db***" against "***IMovieDb***" as shown below.

```
private IMovieDb _movieDb;
```

```
        _movieDb = moviedb;
    }
     [OutputCache(CacheProfile= "Short", Location = OutputCacheLocation.Server,
VaryByHeader = "Accept-Language")]
    public ActionResult Index(string SearchQuery = null,int page=1)
    {

        var model = _movieDb.Query<Movie>()
                    .OrderByDescending(m => m.reviews.Count())
              .Where(m=>SearchQuery==null||m.MovieName.StartsWith(SearchQuery))
                    .Select(m => new MovieViewModel
                    {
                        Id = m.Id,
                        MovieName = m.MovieName,
                        DirectorName = m.DirectorName,
                        ReleaseYear = m.ReleaseYear,
                        NoOfReviews = m.reviews.Count()
                    }).ToPagedList(page,10);
        if (Request.IsAjaxRequest())
        {
            return PartialView("_movies", model);
        }
        return View(model);
    }
```

So, as you can see in the above snippet,

that I have created two constructors. One is the default constructor which will initialize this to "***MovieDb***" which we have been using in our application. And another one is the parameterized constructor where in I'll pass the **Interface** reference. Now, this will give a chance to **Unit Test** to pass. Now, in the test I need to setup a **Fake Database**. But, before that I need to create one **Fake Class**. So, below is the Fake class which I have created in its finished form.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MovieReview.Models;

namespace MovieReview.Tests.UnitTests
{
    public class FakeMovieDb : IMovieDb
    {
        public IQueryable<T> Query<T>() where T : class
        {

            return Sets[typeof (T)] as IQueryable<T>;
        }

        public void Dispose()
        {

        }

        public void AddSet<T>(IQueryable<T> objs)
        {
            Sets.Add(typeof(T),objs);
        }

        public Dictionary<Type,object> Sets = new Dictionary<Type, object>();
    }
}
```

Now, let me explain what this is doing

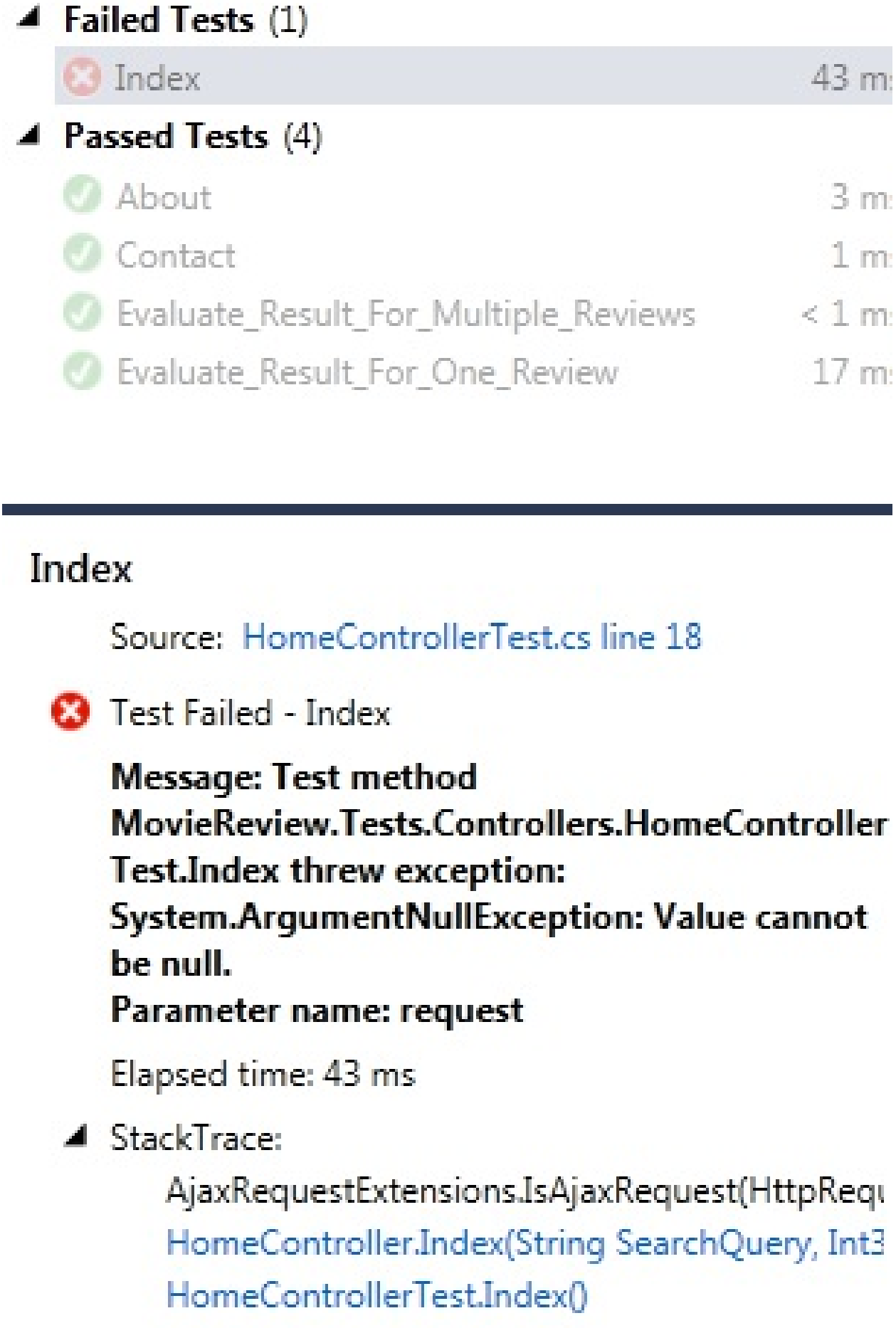
here. So, behind the scene it's just creating the in memory data. So, here 1st thing is that I have defined one dictionary of "***Type***". So, when you want to query a movie, will pull out that movie out of that dictionary by using "***IQueryable***" and pass it on to the Test. Now, there is also a method "***AddSet***" which allows you to add a set which you want inside this **Fake Database**. Now, for this to work I have also added a class which will add some 200 movies and every movie with a rating 5 as shown below.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MovieReview.Models;

namespace MovieReview.Tests.UnitTests
{
    class TestData
    {
        public static IQueryable<Movie> Movies
        {
            get
            {
                var movies = new List<Movie>();
                for (int i = 0; i < 200; i++)
                {
                    var movie = new Movie();
                    movie.reviews = new List<MovieReviews>()
                    {

                        new MovieReviews{ReviewerRating = 5}
                    };
                }
                return movies.AsQueryable();
            }
        }
    }
}
```

Now, with the above piece of code in place, when I'll run the same it will again fail, but now it fails for a different reason. This time it fails for Ajax request which you can see below in the screen shot of the Test Explorer.


Failed Tests (1)
Index 43 m
Passed Tests (4)
About 3 m
Contact 1 m
Evaluate_Result_For_Multiple_Reviews < 1 m
Evaluate_Result_For_One_Review 17 m
Index
Source: HomeControllerTest.cs line 18
Test Failed - Index
Message: Test method MovieReview.Tests.Controllers.HomeControllerTest.Index threw exception: System.ArgumentNullException: Value cannot be null.
Parameter name: request
Elapsed time: 43 ms
StackTrace:
AjaxRequestExtensions.IsAjaxRequest(HttpReq
HomeController.Index(String SearchQuery, Int3
HomeControllerTest.Index()

Now, to overcome this, I have another class defined which is "***FakeControllerClass***" which will give a **Fake Context** basically a **Fake HTTP Context** which will have a **Fake Request Object**. Now, this Fake Request object is really doing nothing. It is just returning null and empty collection everywhere.

```
using System;
using System.Collections.Generic;
using System.Collections.Specialized;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;

namespace MovieReview.Tests.UnitTests
{
   class FakeController :ControllerContext
    {
        private HttpContextBase _ctx = new FakeHttpContext();
```

```
     public override System.Web.HttpContextBase HttpContext
     {
         get
         {
             return _ctx;
         }
         set
         {
             _ctx = value;
         }
     }
 }

class FakeHttpContext : HttpContextBase
 {
     HttpRequestBase _request = new FakeHttpRequest();

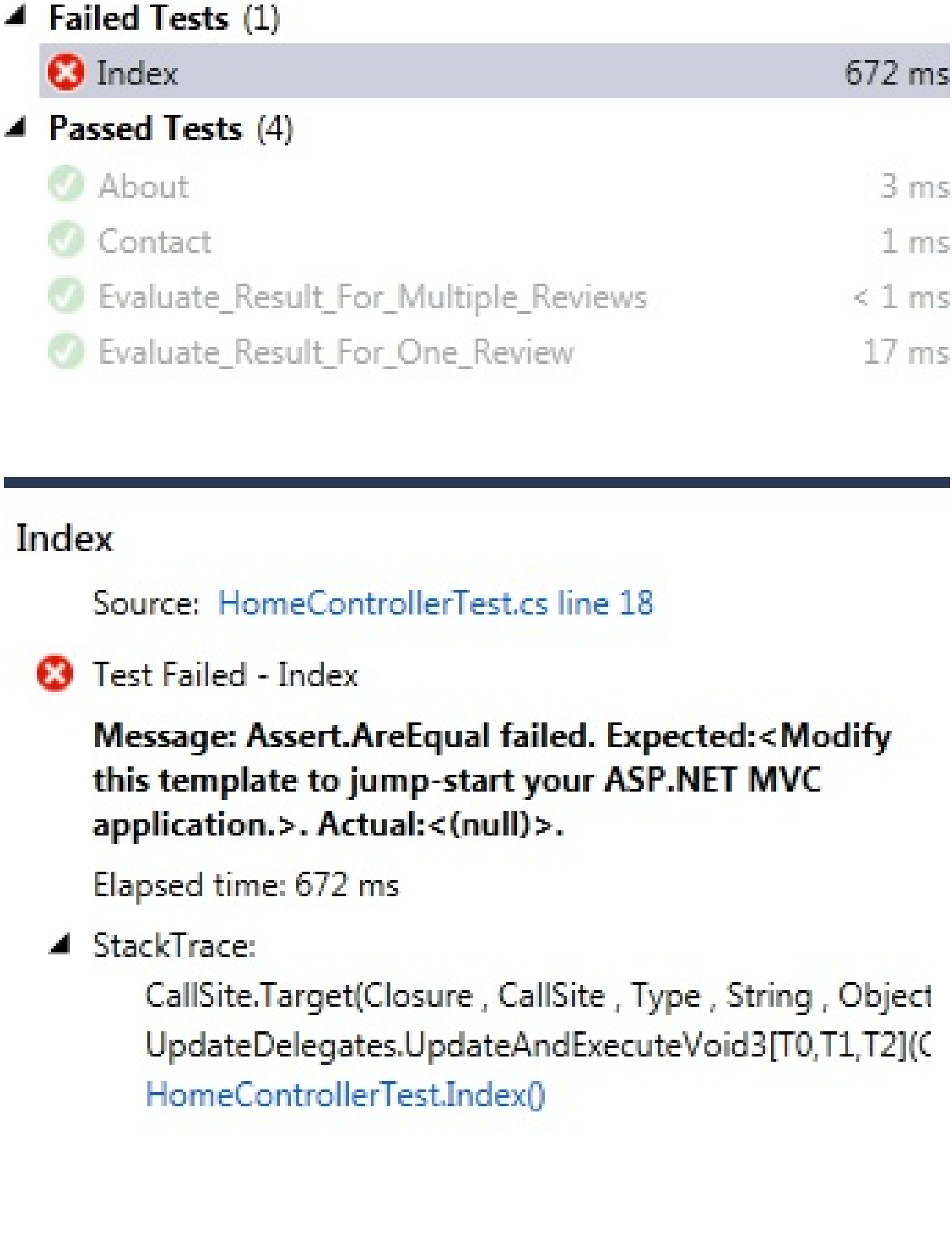
     public override HttpRequestBase Request
     {
         get
         {
             return _request;
         }
     }
 }

class FakeHttpRequest : HttpRequestBase
{
    public override string this[string key]
    {
        get
        {
            return null;
        }
    }

    public override NameValueCollection Headers
    {
        get
        {
            return new NameValueCollection();
        }
    }
}
```

```
}
```

Now, with this code in place, when I run the same, it will again fail, but this time at least it came to assert statement as shown below.


Failed Tests (1)
Index 672 ms
Passed Tests (4)
About 3 ms
Contact 1 ms
Evaluate_Result_For_Multiple_Reviews < 1 ms
Evaluate_Result_For_One_Review 17 ms
Index
Source: HomeControllerTest.cs line 18
Test Failed - Index
Message: Assert.AreEqual failed. Expected:<Modify this template to jump-start your ASP.NET MVC application.>. Actual:<(null)>.
Elapsed time: 672 ms
StackTrace:
CallSite.Target(Closure , CallSite , Type , String , Object
UpdateDelegates.UpdateAndExecuteVoid3[T0,T1,T2](
HomeControllerTest.Index()

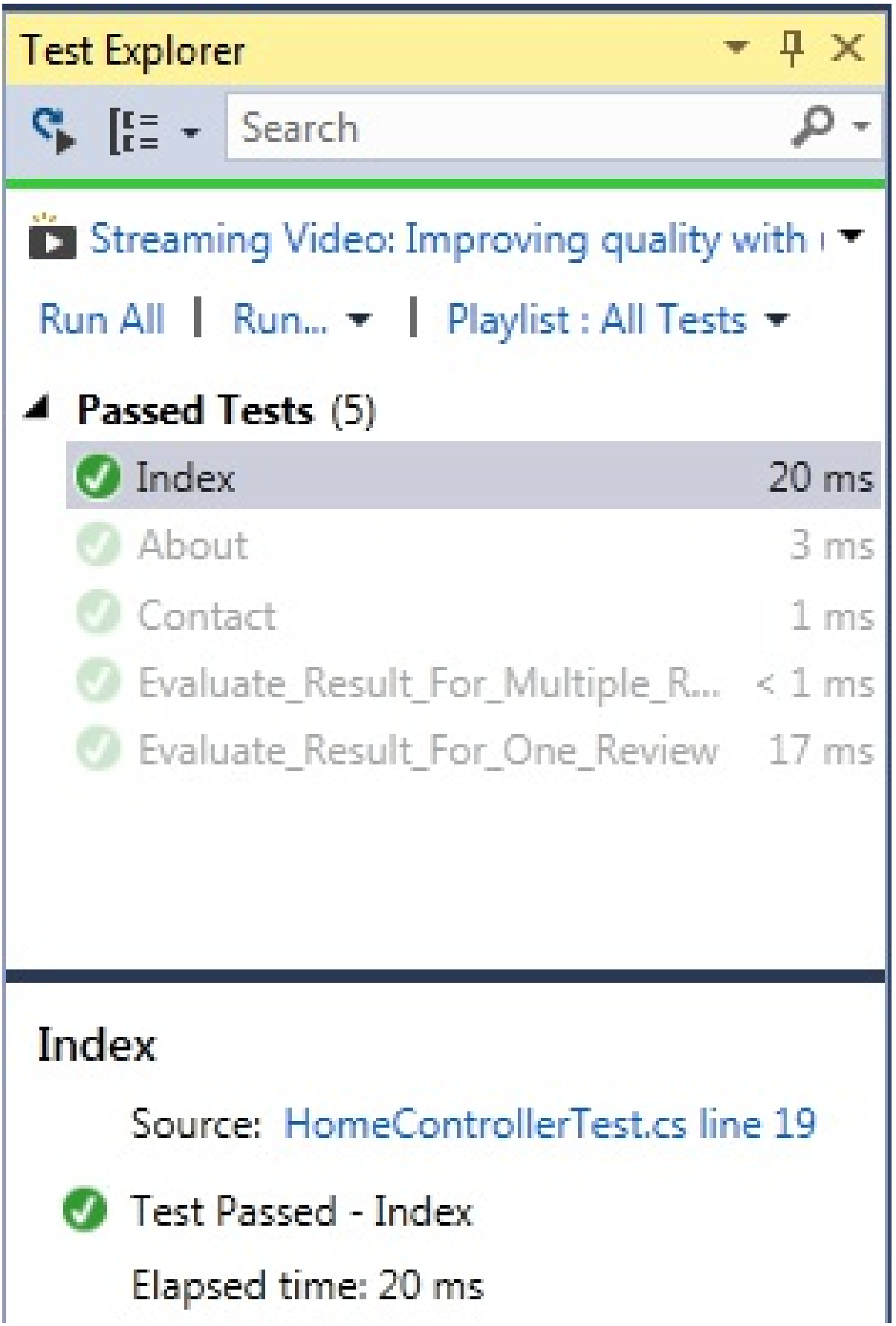
Now, the assert is failing for obvious reasons, because now view bag is empty. And also it matching against the string "***Modify this template to jump-start your ASP.NET MVC application***."; which we have changed. So, basically this assert itself is not valid in this context. So, now I need to modify this. Now, what I am going assert is; just check whether the model is null or not.

```
[TestMethod]
        public void Index()
        {
            var db = new FakeMovieDb();
            db.AddSet(TestData.Movies);
            // Arrange
            HomeController controller = new HomeController(db);
```

```
    controller.ControllerContext = new FakeController();
    // Act
    ViewResult result = controller.Index() as ViewResult;

    // Assert
    Assert.IsNotNull(result.Model);
}
```

Now, with the above code in place, I should be able to pass this test which you can see below in the screen shot.

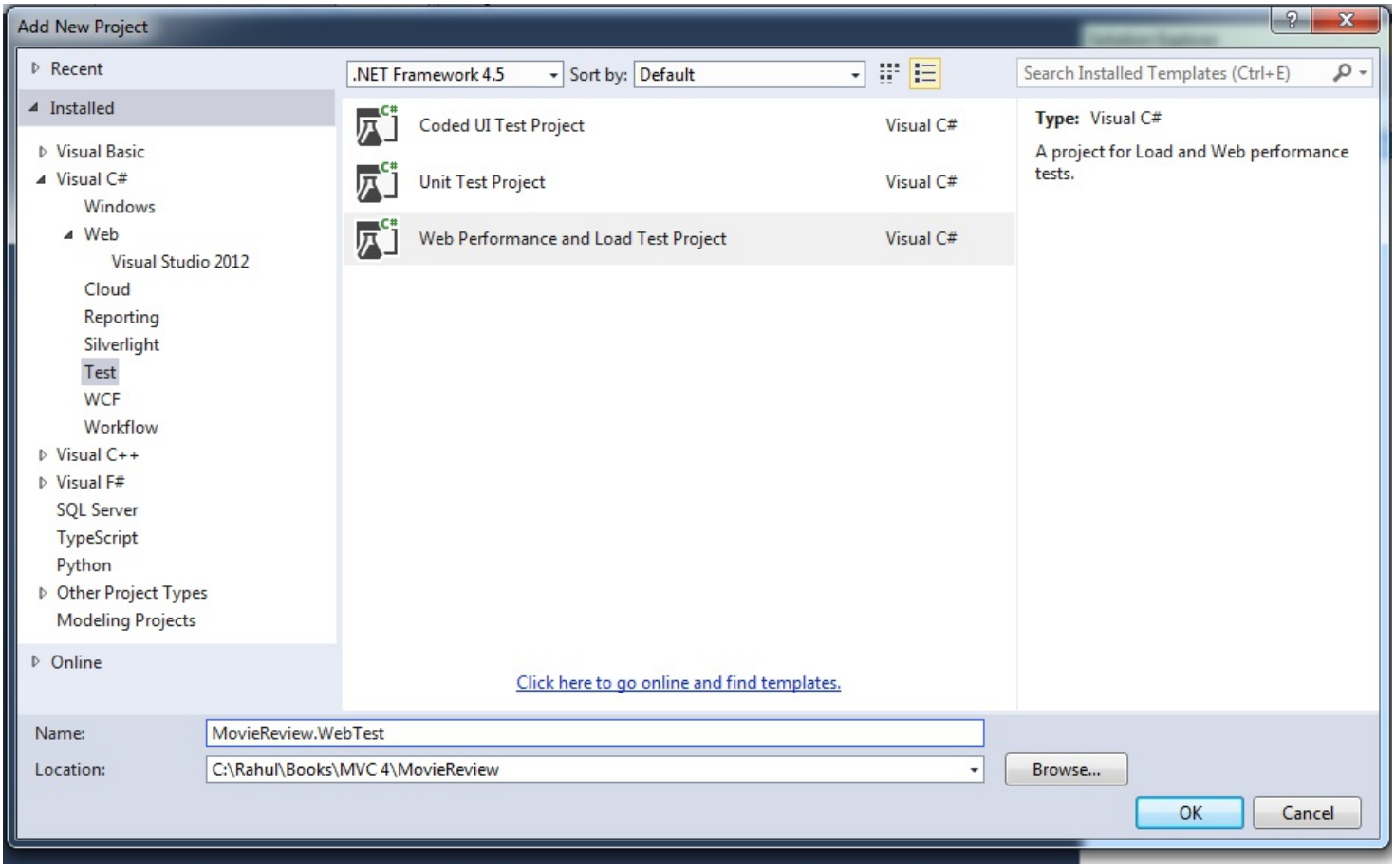

Test Explorer
Search
Streaming Video: Improving quality with
Run All | Run... | Playlist : All Tests
Passed Tests (5)
Index 20 ms
About 3 ms
Contact 1 ms
Evaluate_Result_For_Multiple_R... < 1 ms
Evaluate_Result_For_One_Review 17 ms
Index
Source: HomeControllerTest.cs line 19
Test Passed - Index
Elapsed time: 20 ms

So, this way we can create as many test cases as we want. So, I leave this on readers that how many test cases they would like to test for the application logic.

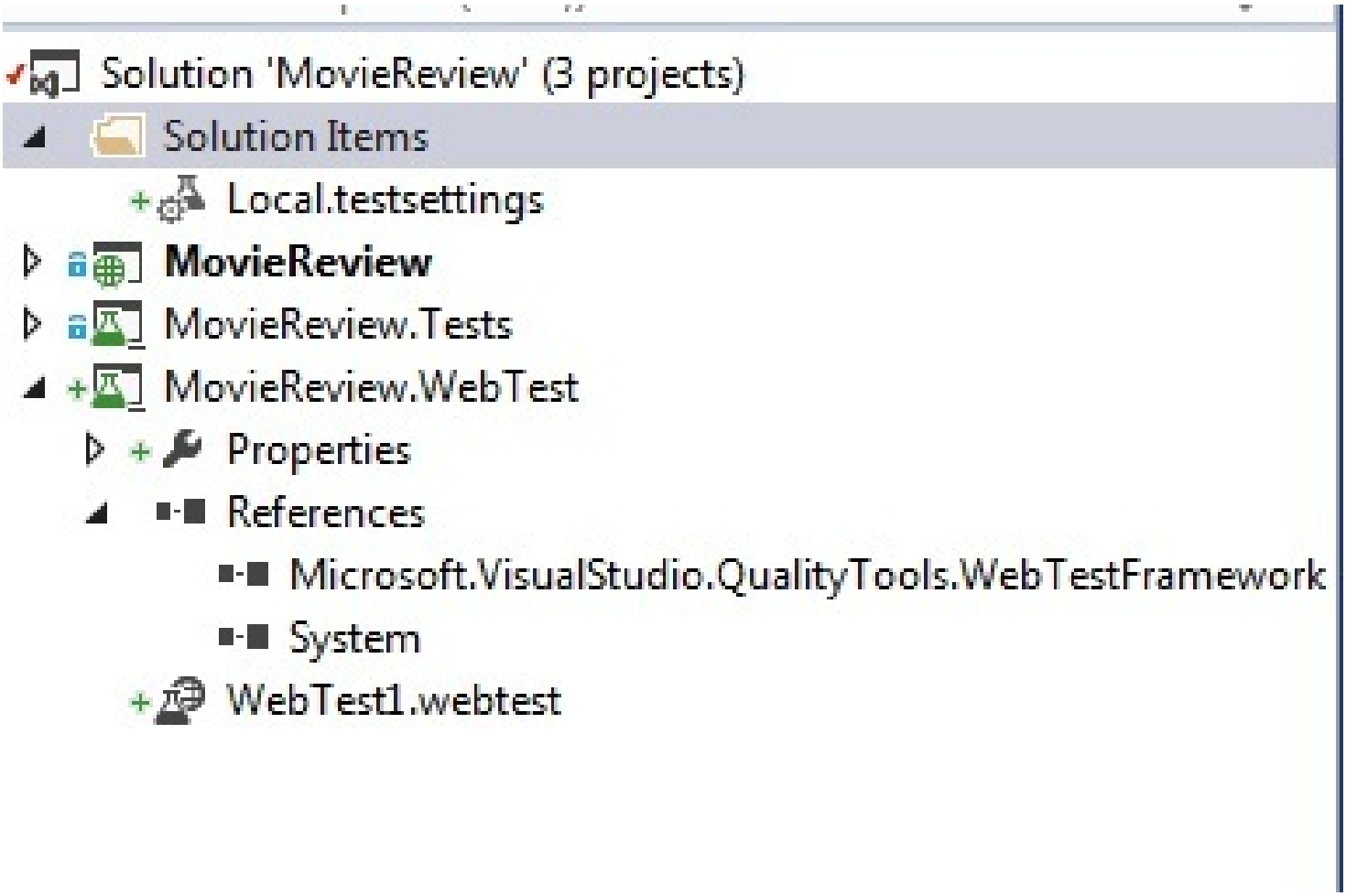
Web Tests:-

Now, Unit Tests basically enable you to test pieces of your application in which they validate the logic within the application and make sure that everything is correct. However, there are other aspects which couldn't get checked with these unit tests but they are equally important. Like there could be certain code flaws in your application

which is not capable of handling simultaneous user logins or in short heavy traffic. Now, using Visual Studio you can go ahead and configure your Visual Studio for the application in such a way, that it will create virtual users for your application and you can test the application behavior at that instant. So, without wasting time, let's go ahead and add one additional project for Web Test project in our application as shown below.

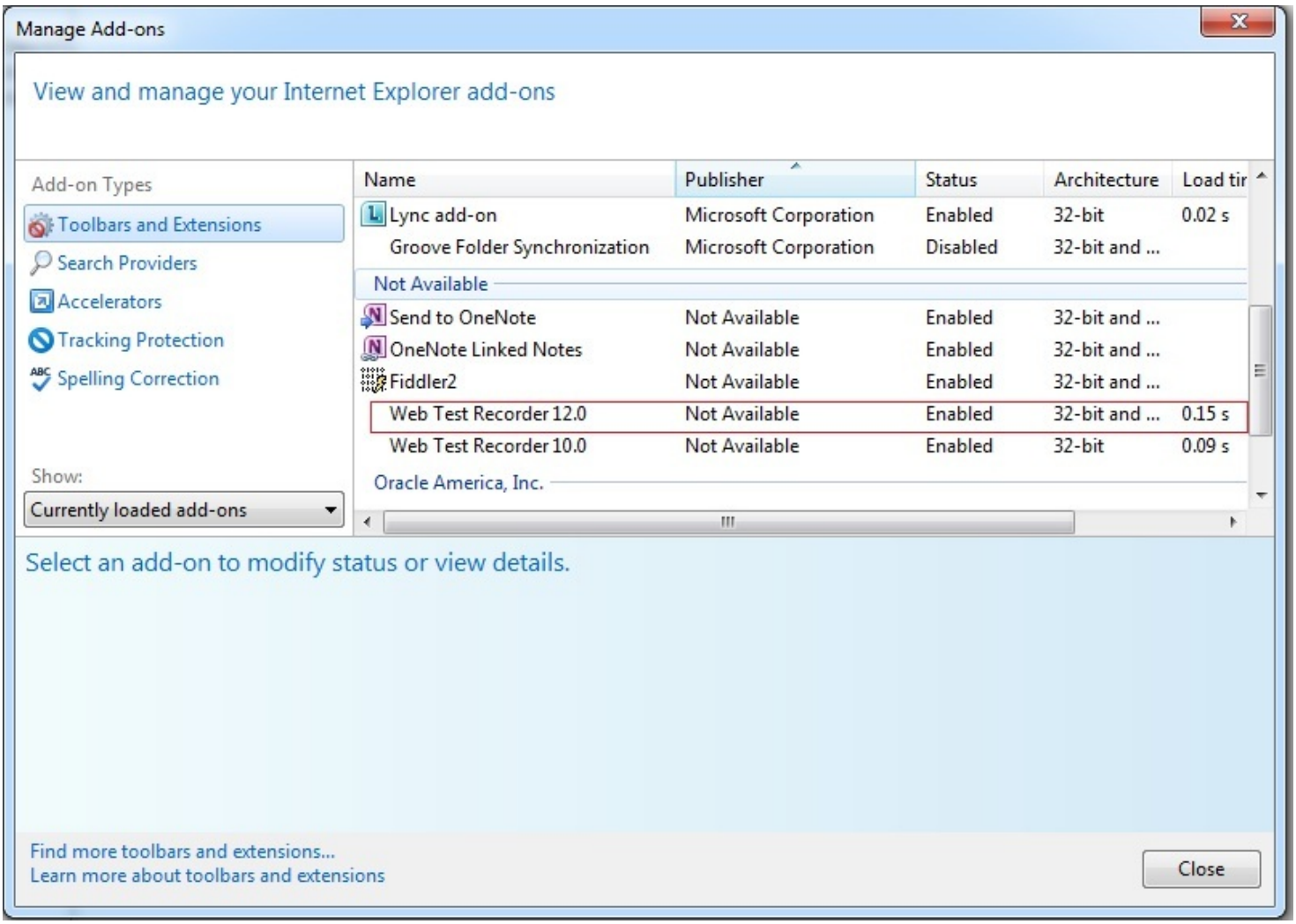


Now, when I click ok, it will create one new project for me in the solution explorer as shown below.



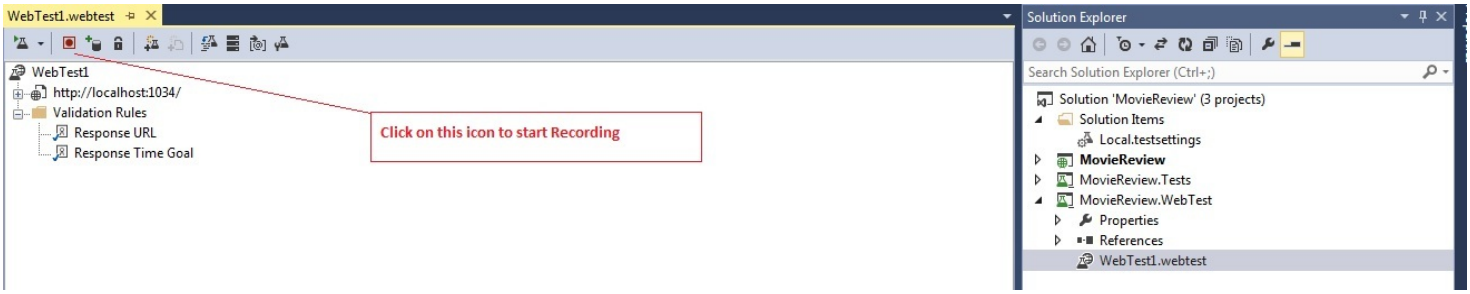
Now, I will delete this default template given to me by visual studio "WebTest1.webtest" and will add a new web test project as shown below. Now, as soon as I add a new web test project, it will automatically open the browser where in it will be recording our sessions as shown below. Also, if by

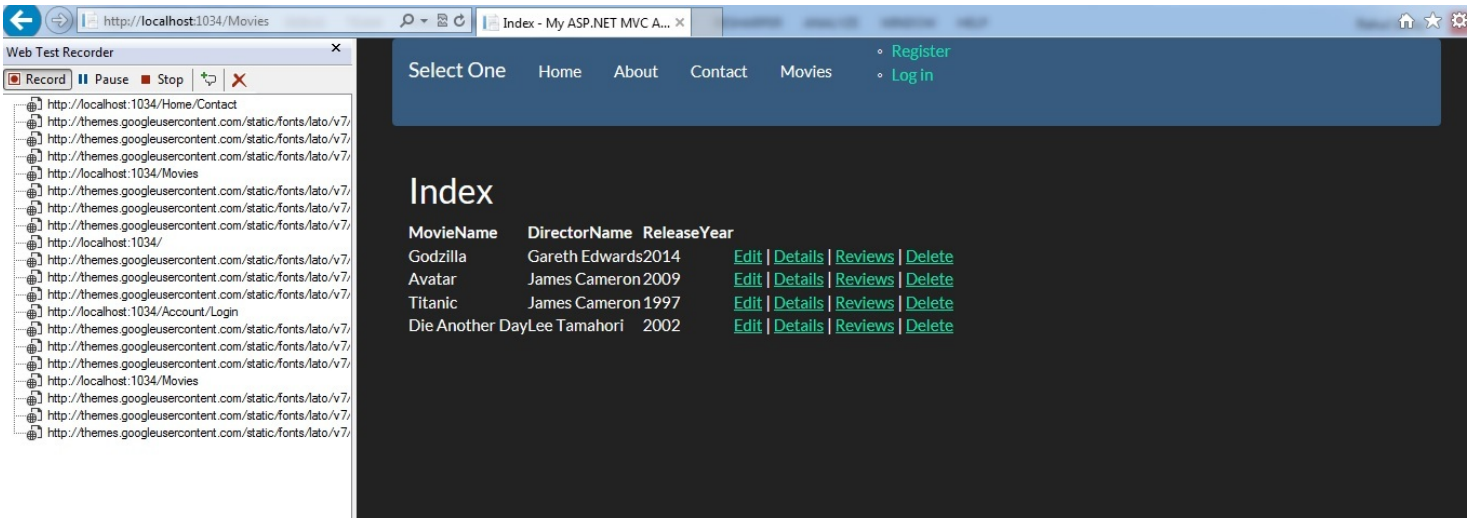
default in your browser web test recorder won't appear check your "***IE Add ONs***" setting and enable it for **Web Test Recorder** as shown below.



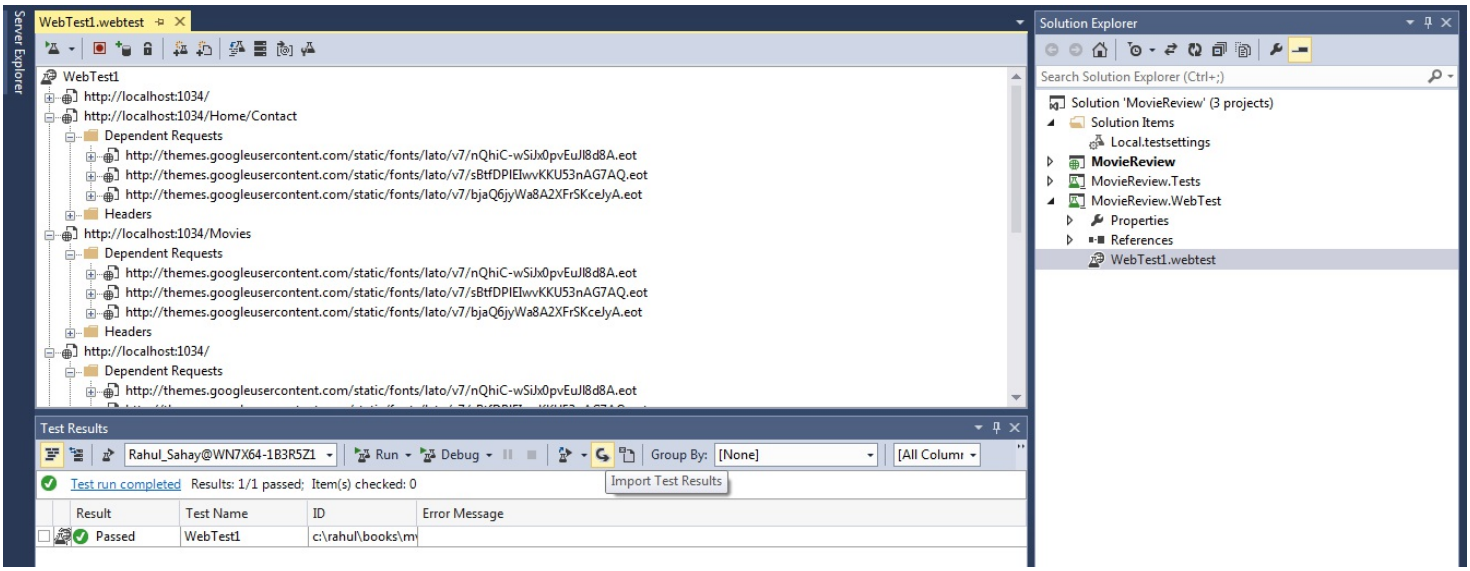
Now, since I have already enabled in my browser so, when I go ahead and click

on below shown icon, it will open the browser with recording bar

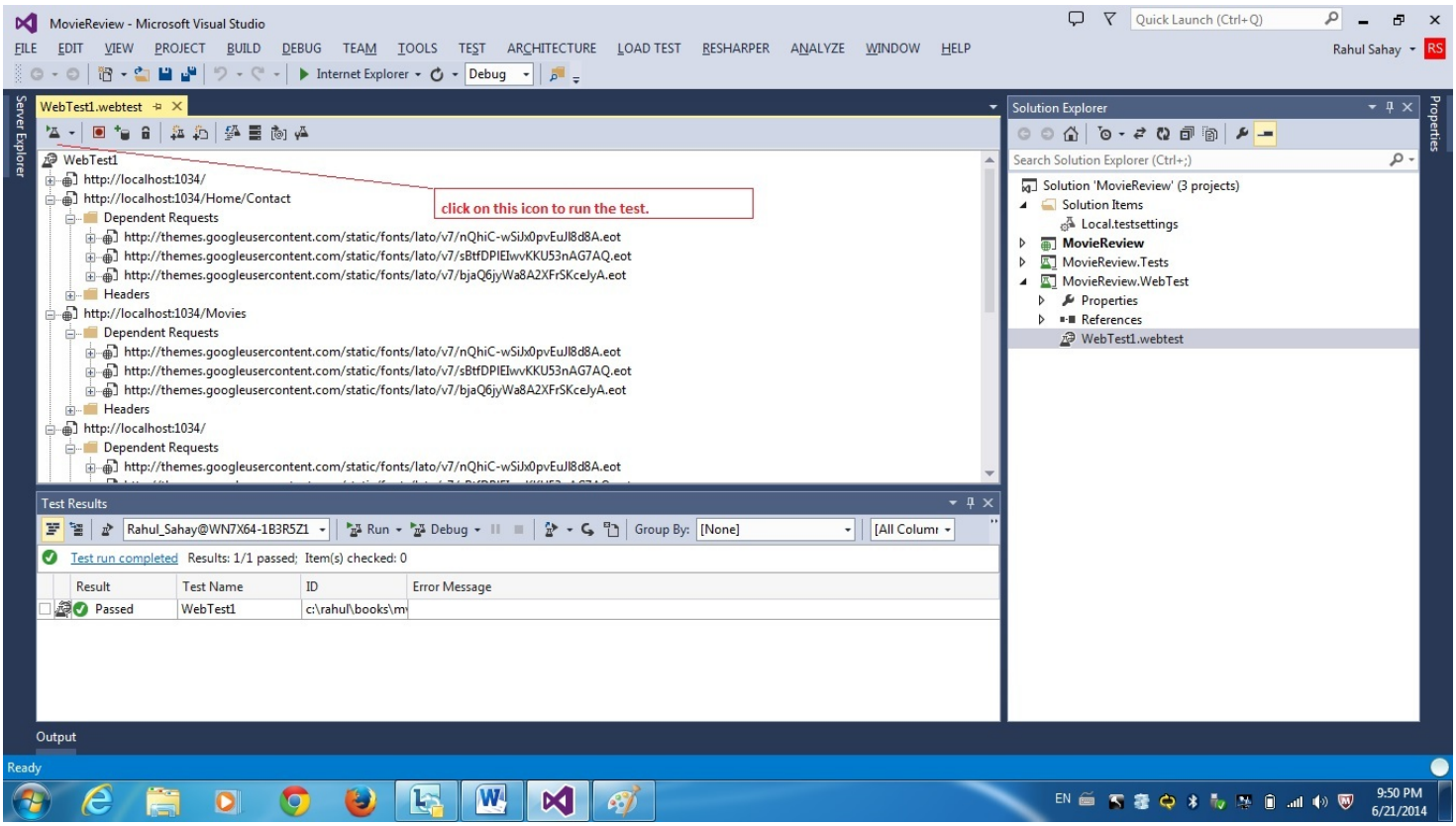




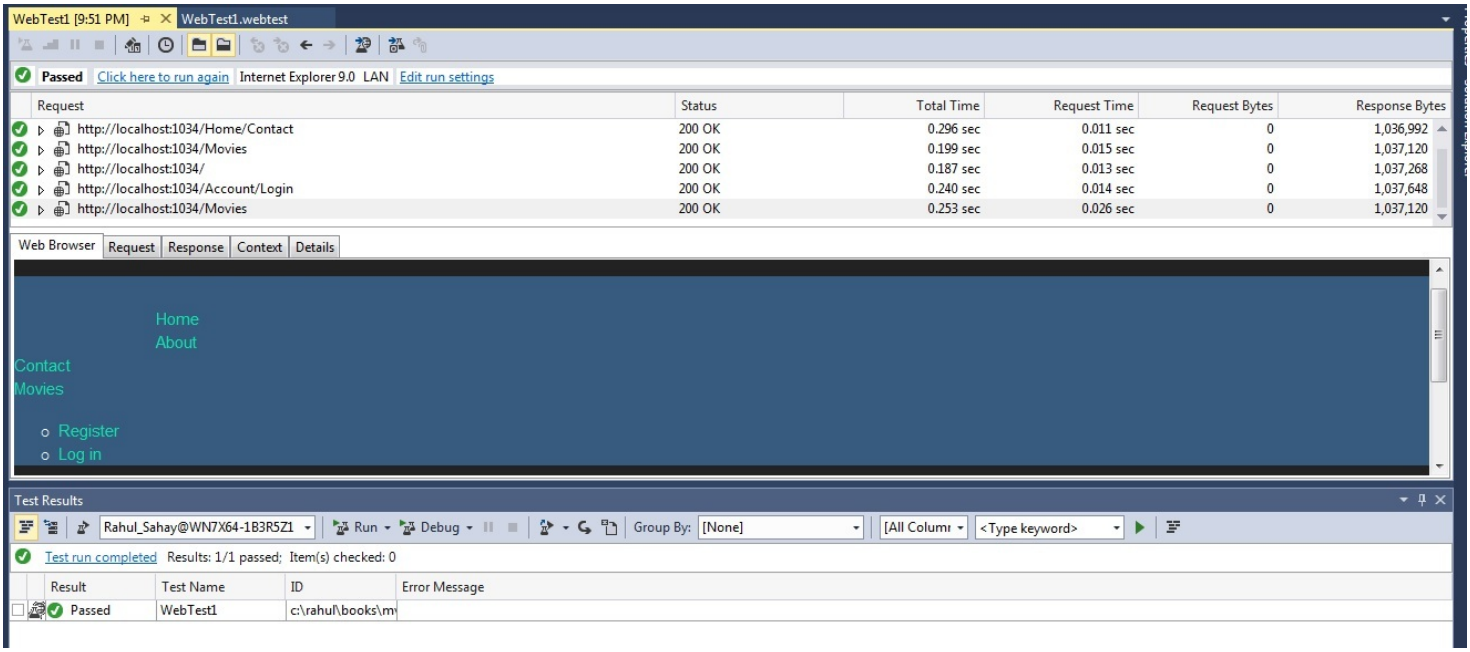
So, as you can see in the left panel, it listed all the http requests when I clicked on various links in my application. Now, when I click on stop recording, it will take me to Visual Studio back with ran results as shown below



Now, when I click on the 1st icon as shown below, then it will run the test and give me snapshot of the Test.



On running the test it will produce the below result.



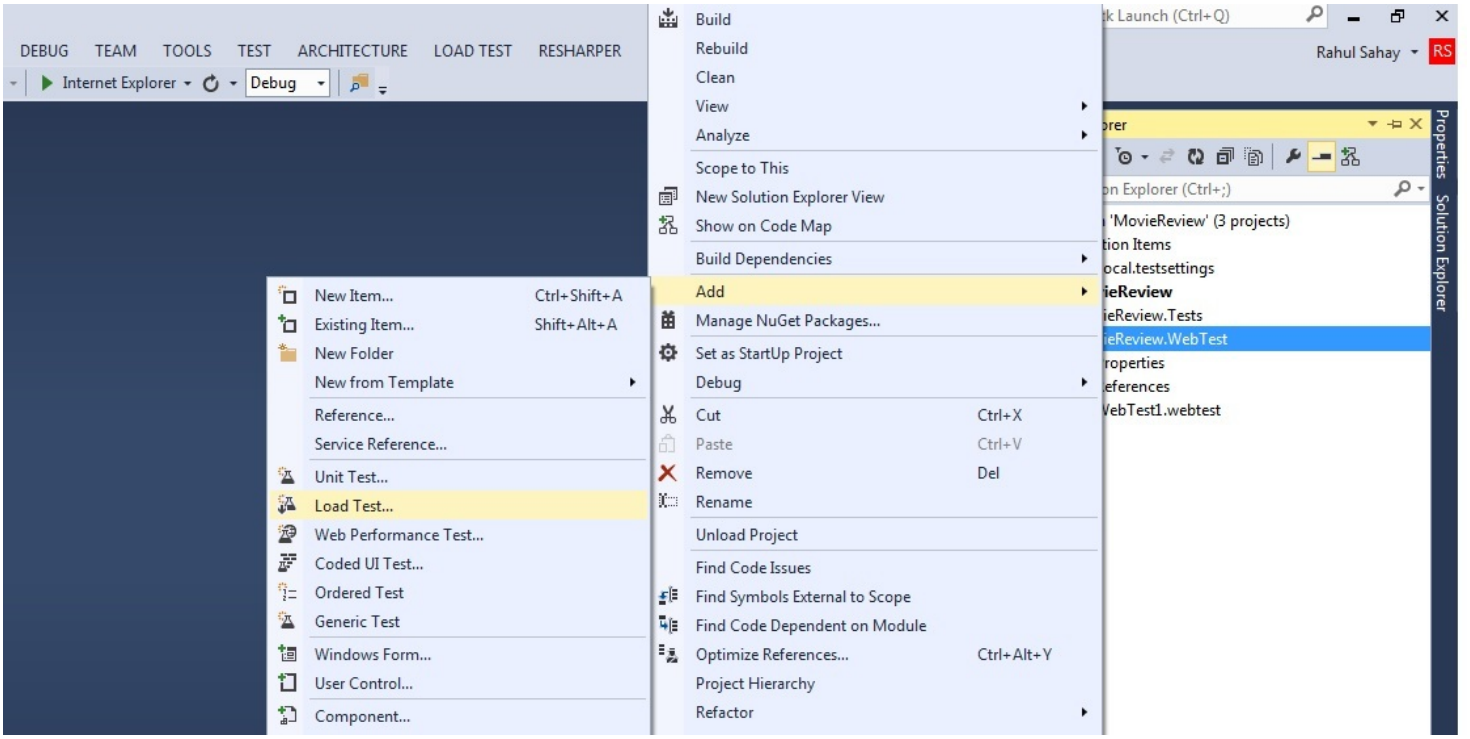
So, as you can see in the above screen shot, it has given page by page analysis, like Status Code, Total Time Taken, Request Time and many more parameters necessary to decide your application performance. Now, like these you can test many other tests like Validation Rule Test, Extraction Rules

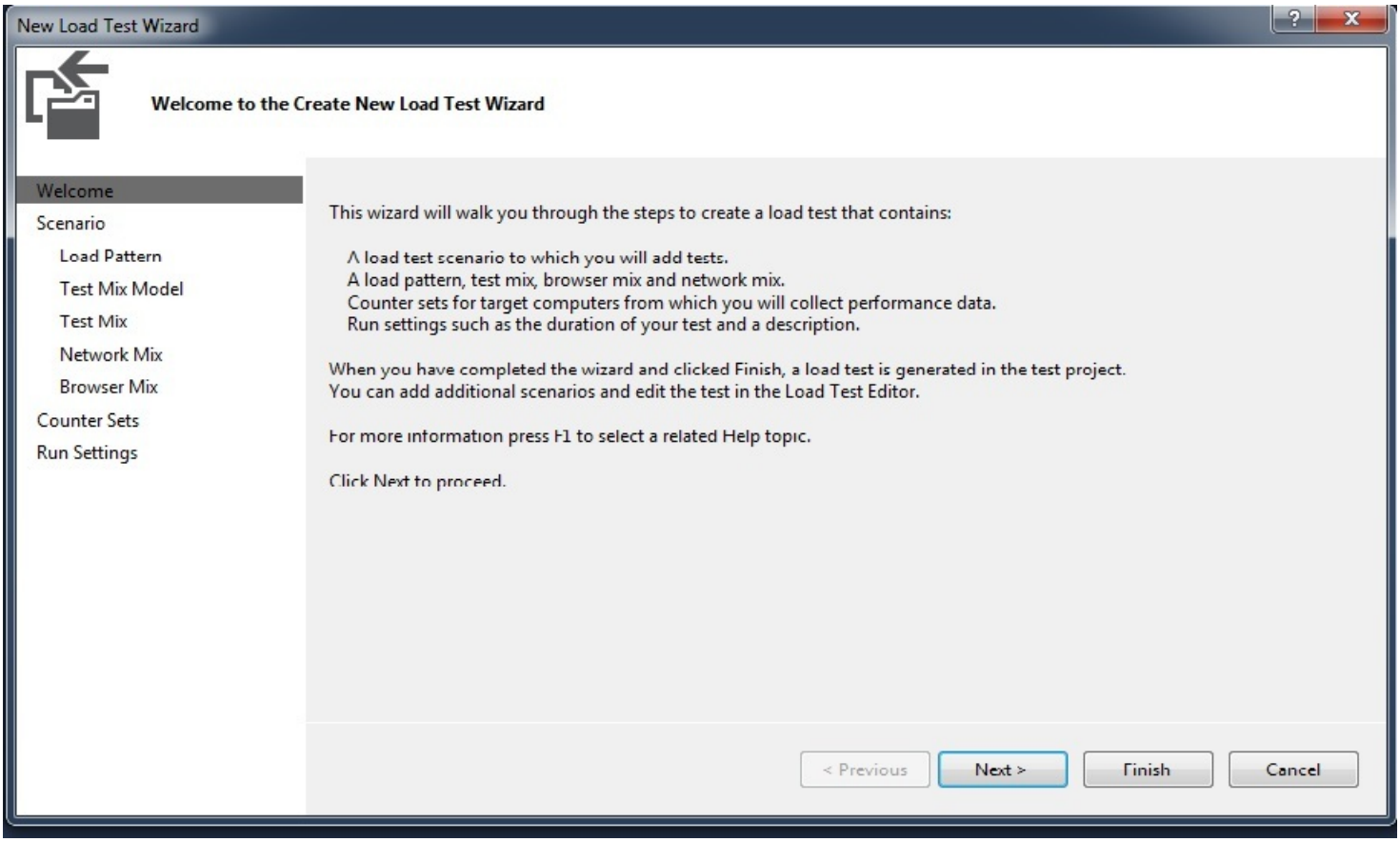
Test and many more.

Load Tests:-

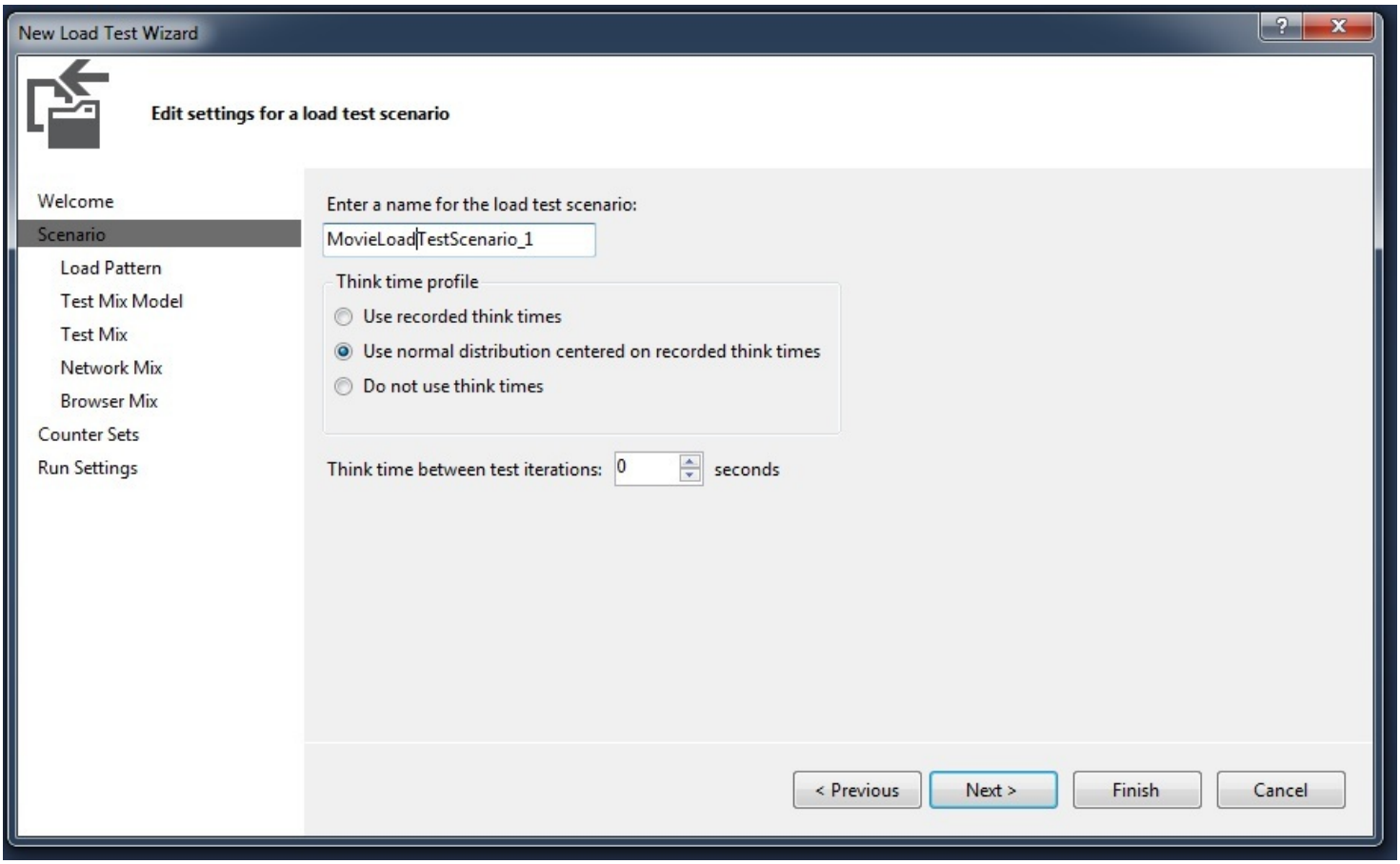
Now, one important piece I would like to demonstrate here is load test. So, basically a **Load Test** is a combination of **Unit Test** and **Web Performance Test**. So, as I said load test is going to simulate your application with lots of virtual users at the same time. So, when you run the load test, some of the most crucial parts to look for are **Performance Counters** like **Perfmon**, **User Statistics** and many more things on the server. So, let's go ahead and add Load Test in our application. So, we

will do right-click on **Web Test Project** and do the following things as shown below in the screen shot

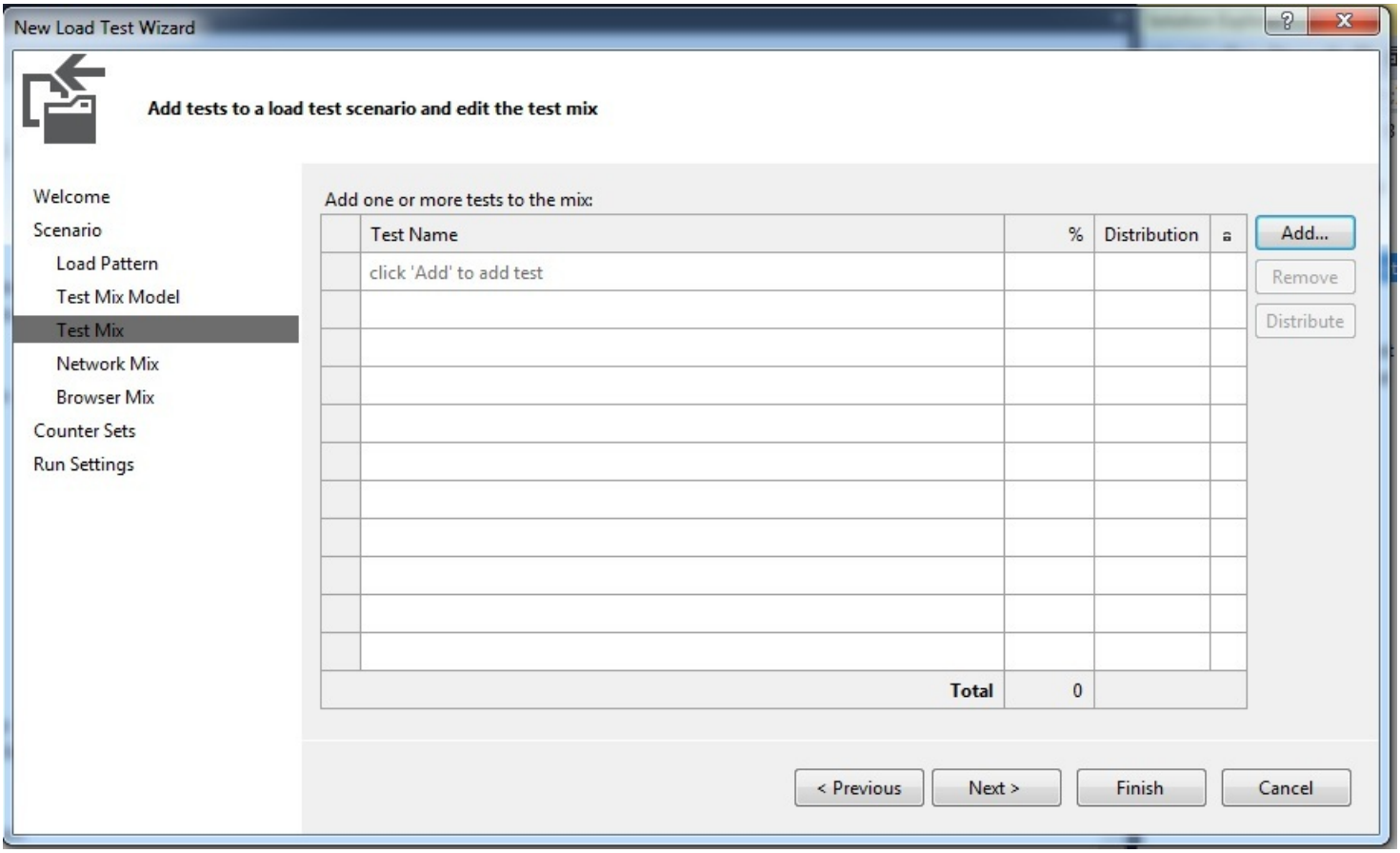




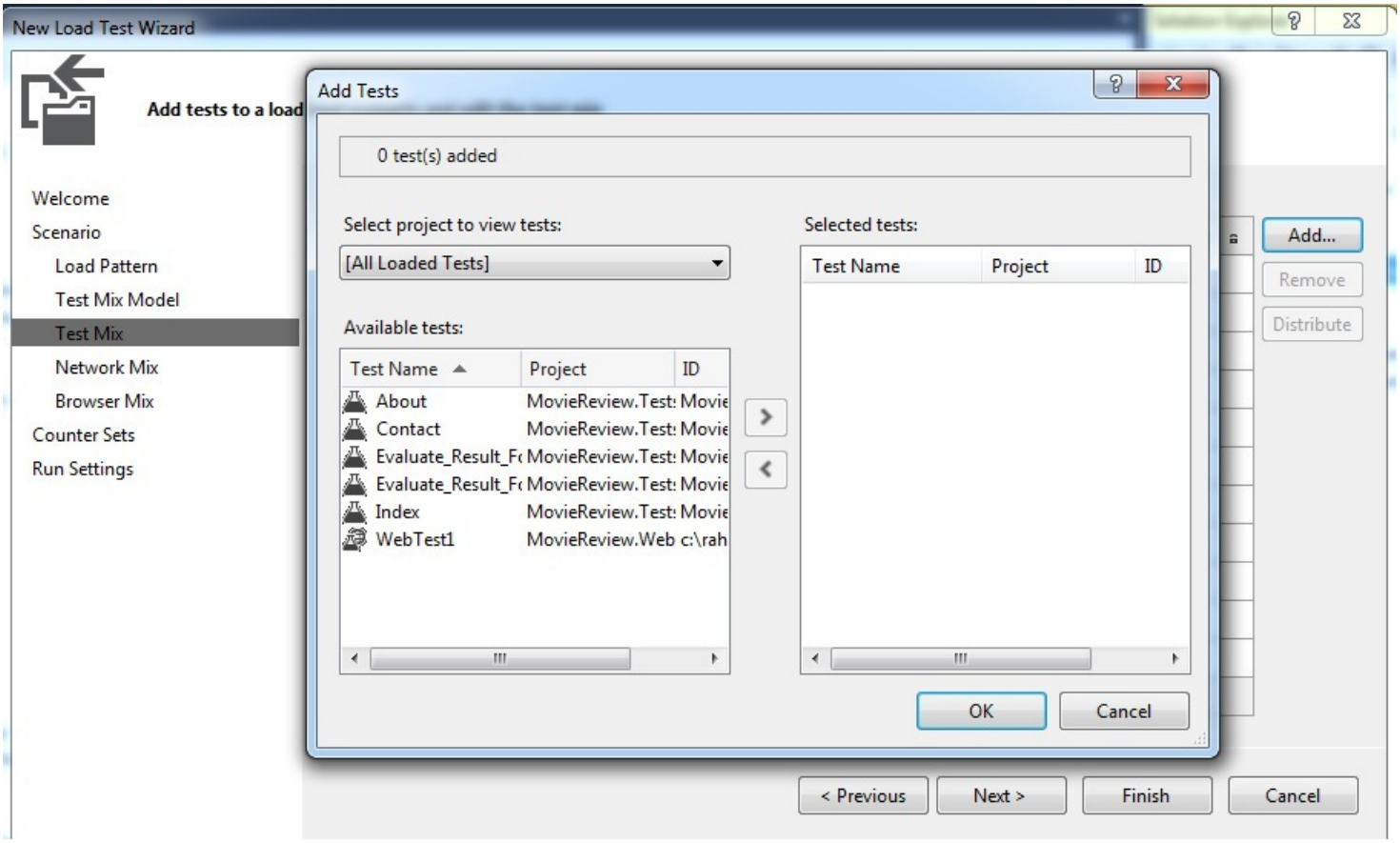
Now, click on next button. This will open a new page where in you can go ahead and set variety of **Test Counters** as shown below.



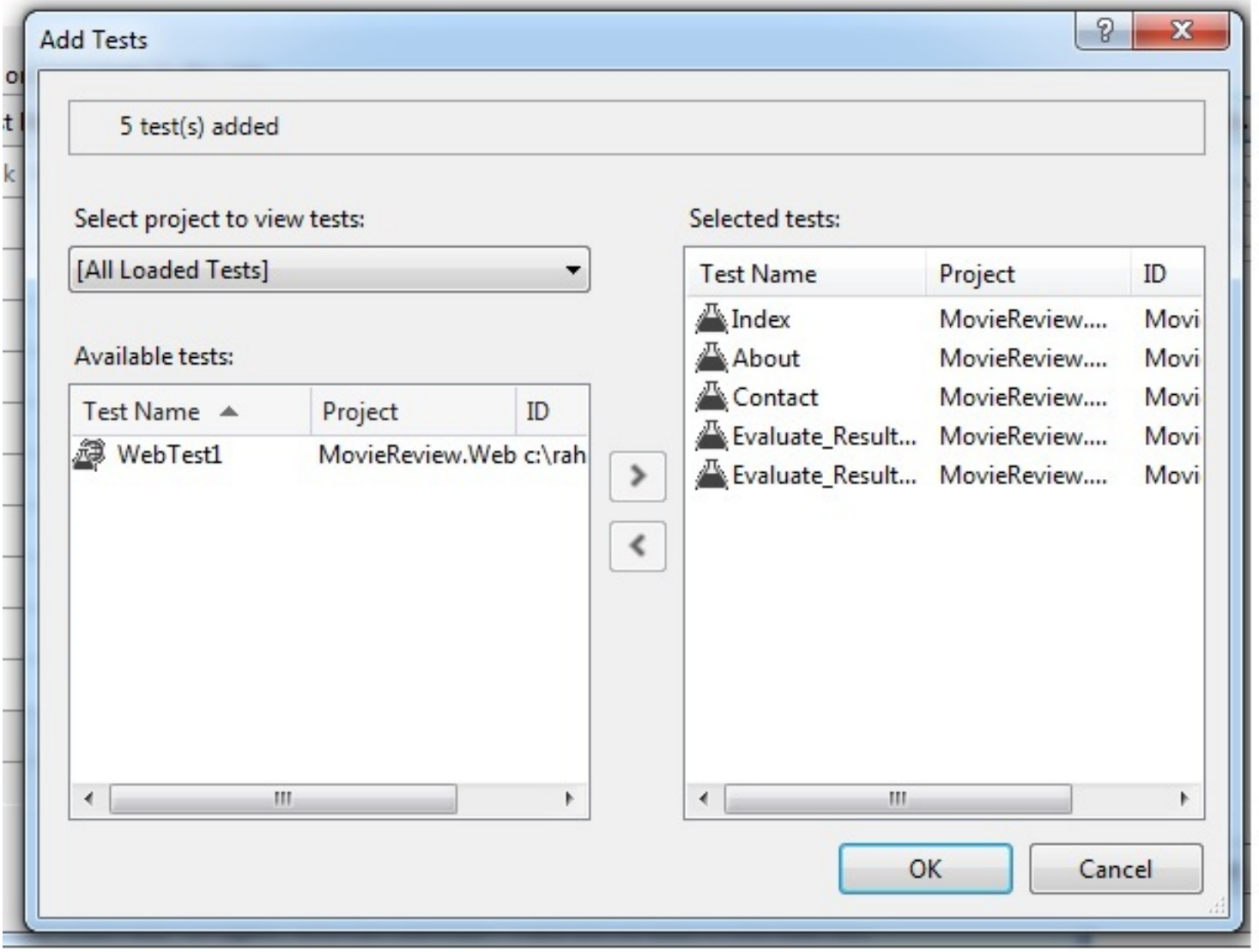
Now, as you can see in the left panel there is variety of test patterns or counters which you can set to test in different scenarios. Currently for demo purpose I am good with default settings, till the point I reach the point **Test Mix** as shown below in the screen shot.



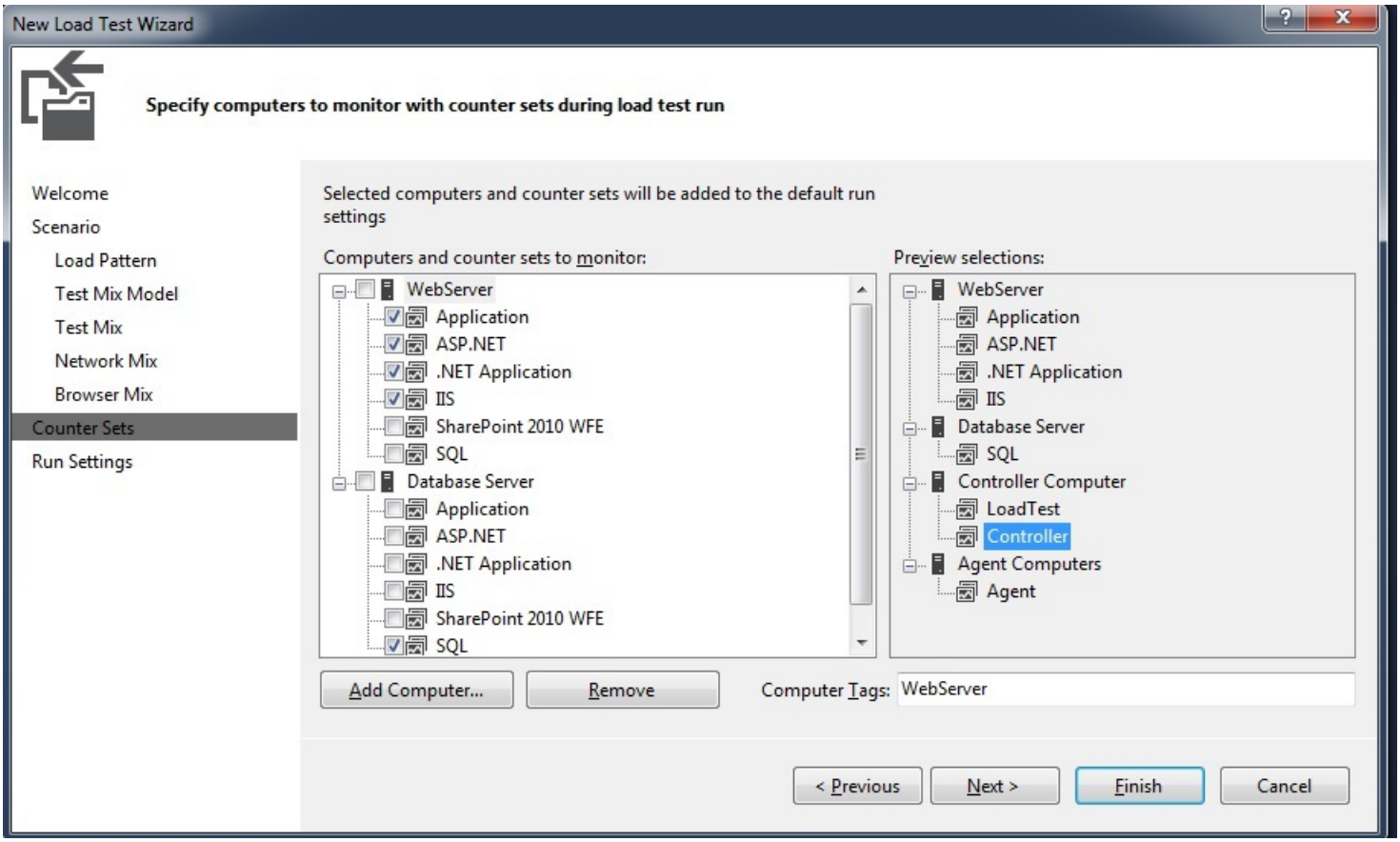
Now, here I need to add test cases by clicking on Add button as shown below in the screen shot.



Now, here you need to select the **Test Cases** which you are interested in by clicking the arrow button.

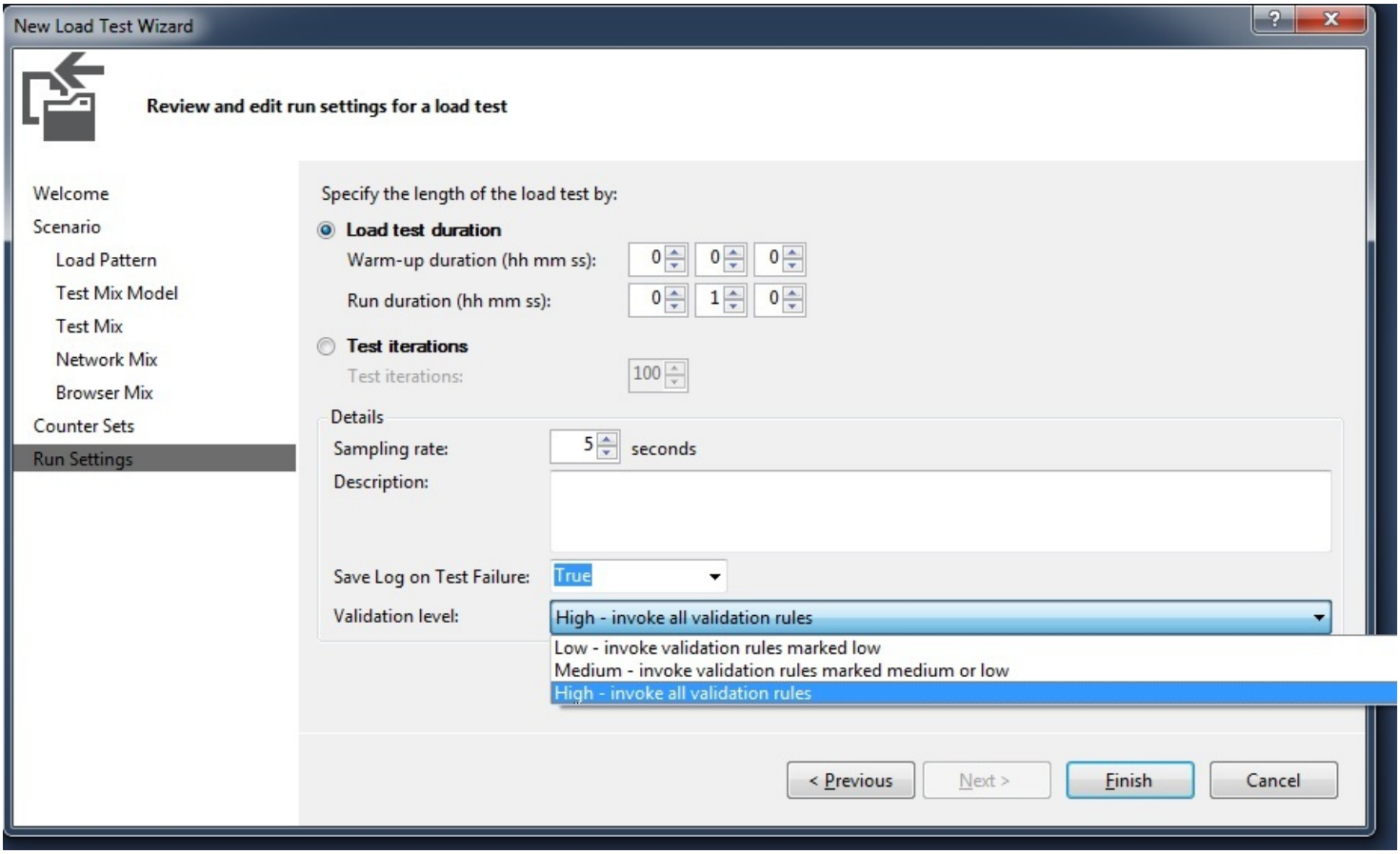


Then I'll click ok and I'll just click on next button till the time I reach "***Counter Sets***". Now, in Counter sets I would like to set couple of things like shown below



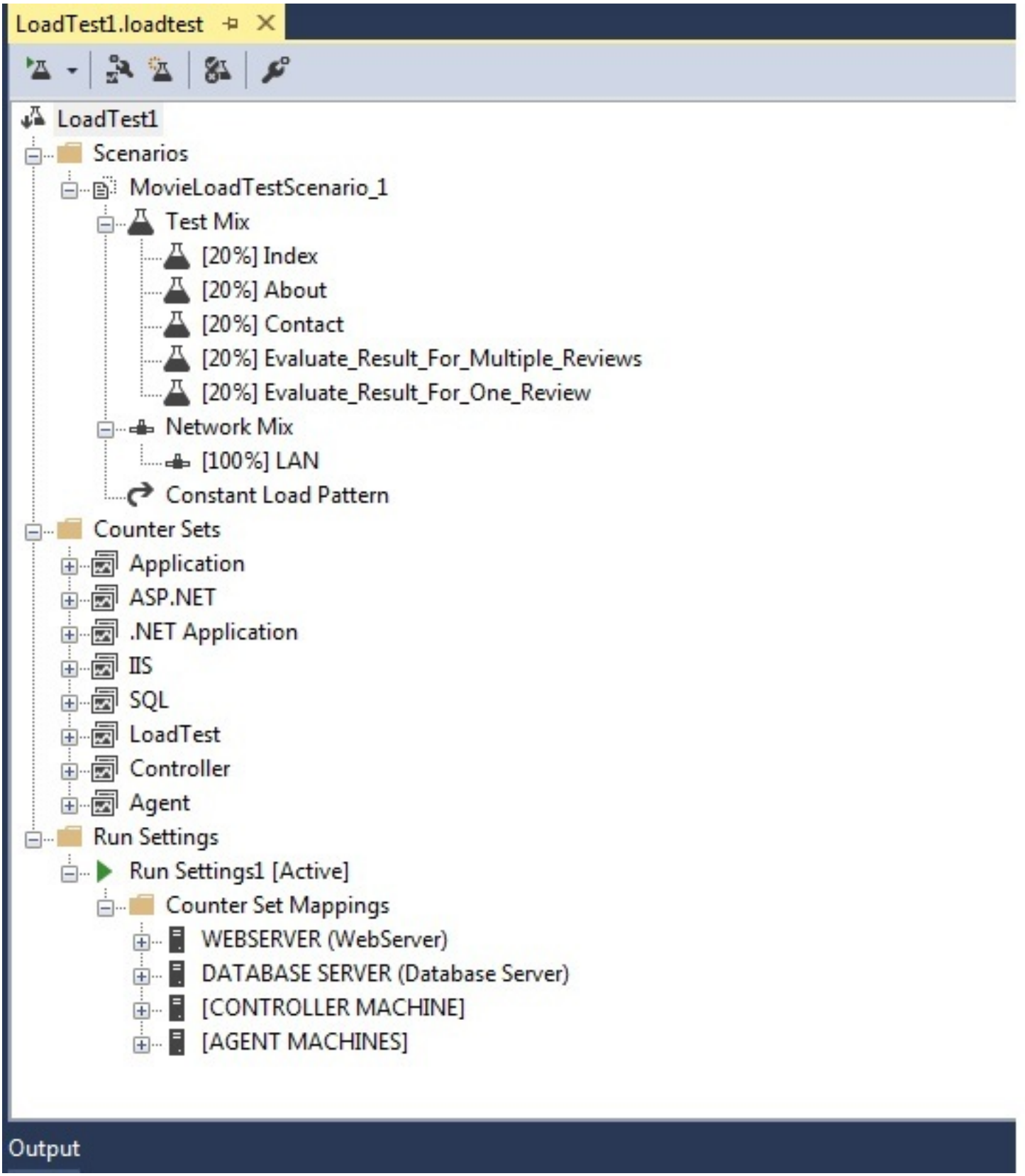
Now, here I have done couple of things, 1st by clicking on Add Computer button I added two nodes, 1st node I have named as "***Webserver***" and added all the server side counters in that, then again I clicked on Add Computer and given the name "***Database***" Server and then added SQL settings in that. If you notice in the above

screen shot, I have also given tag names to both the computers as it would be helpful to analyze when Test is running. Now, in the next page I will set Run Settings as shown below



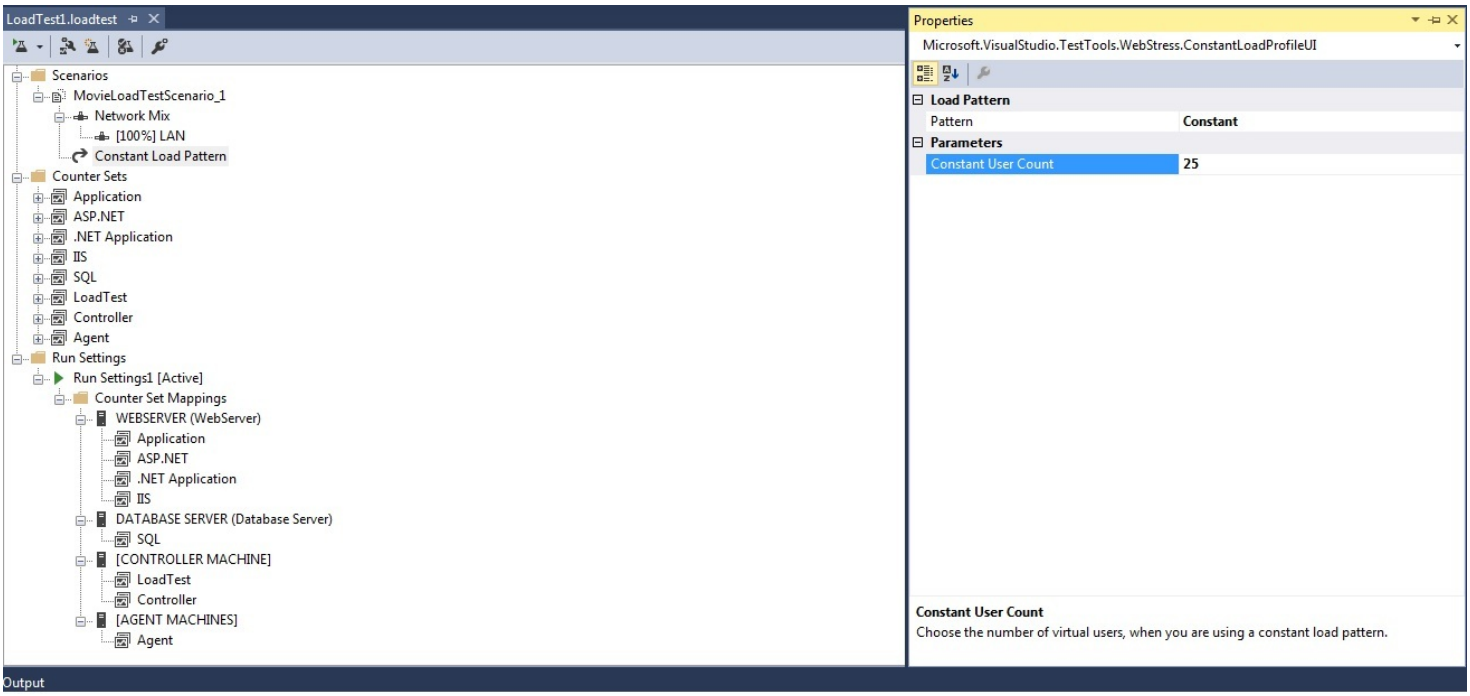
So, here I have set the test duration for 1 minute. Now, this test will run for 1 minute and capture all the data. Other

things like Save the log even if test fails is set to true, that's by default. Also I am keeping Validation rules High, as this is my 1st test so I want all the scenarios should get covered. Now I am going to click on Finish. Once, I click on Finish button, it will generate the Load test for me with all the scenarios and counters which I have set in the wizard as shown below.

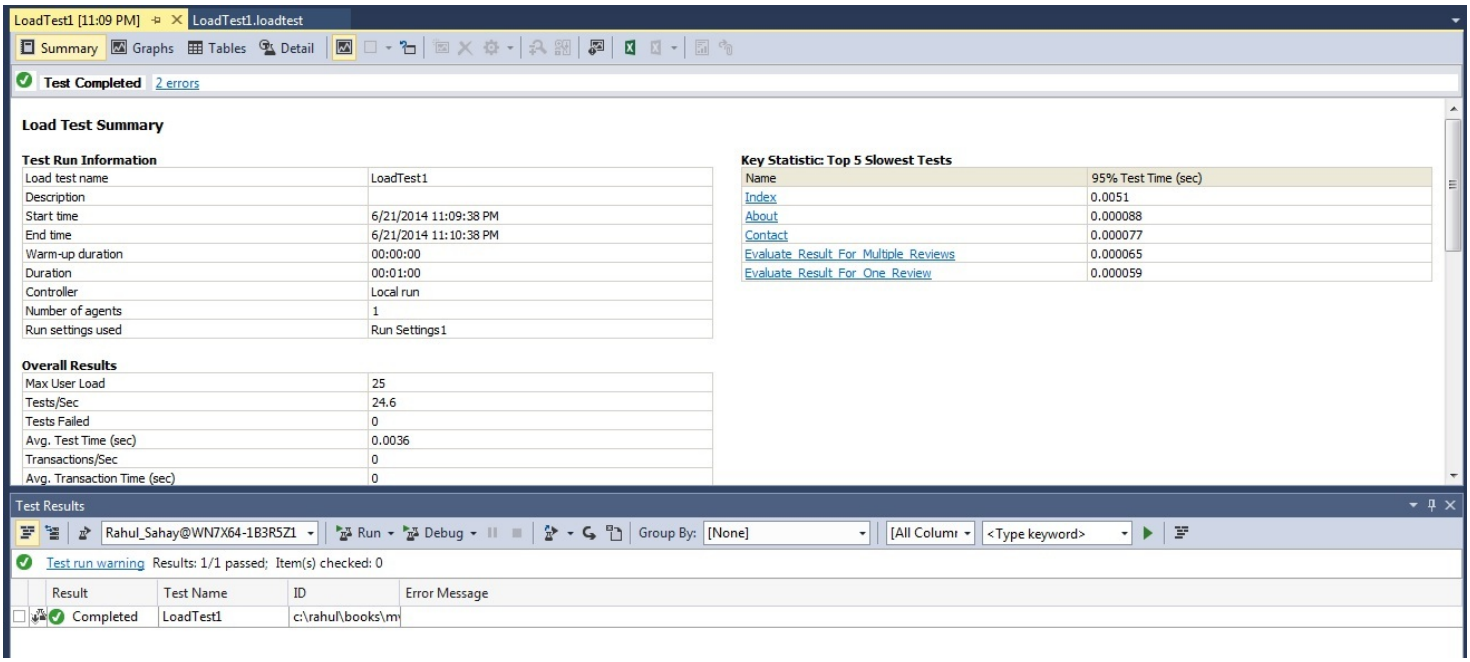


Now, when I check certain properties

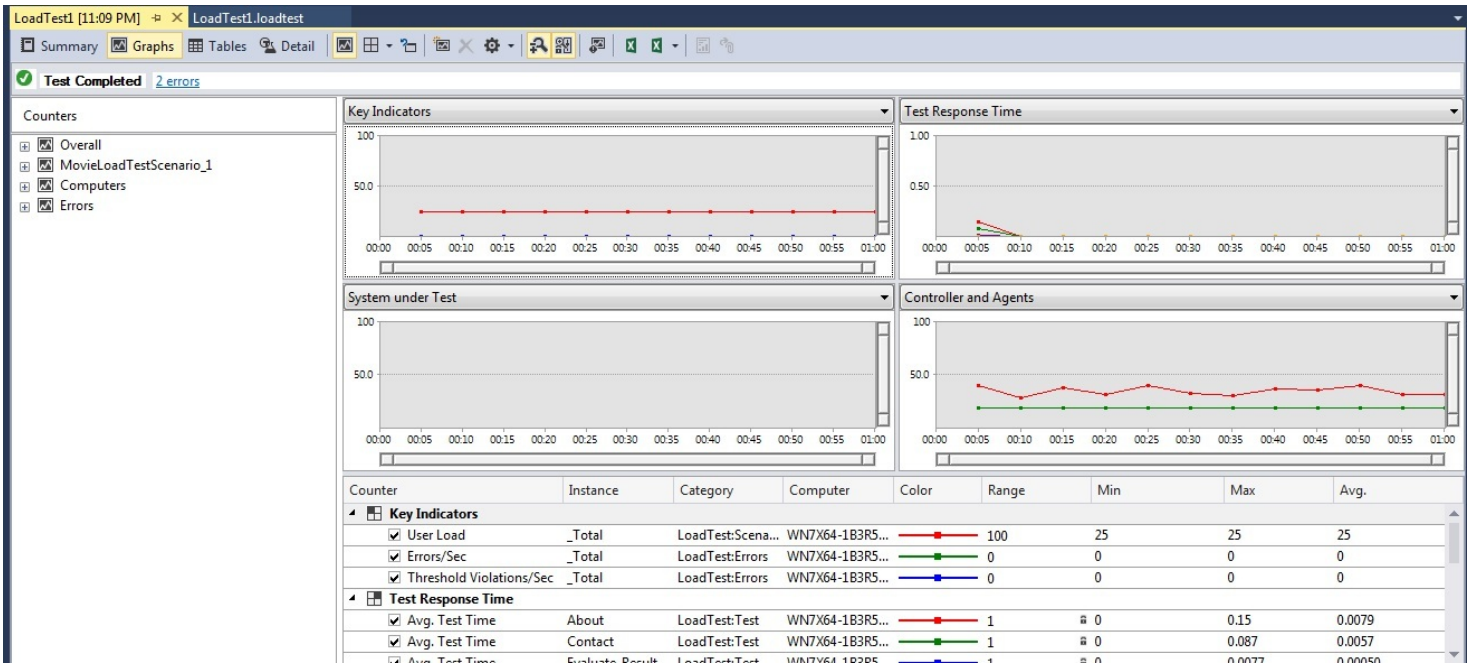
like load pattern property, you will notice that default value is set, and that is set for "***25 users***" which we have chosen from the wizard.



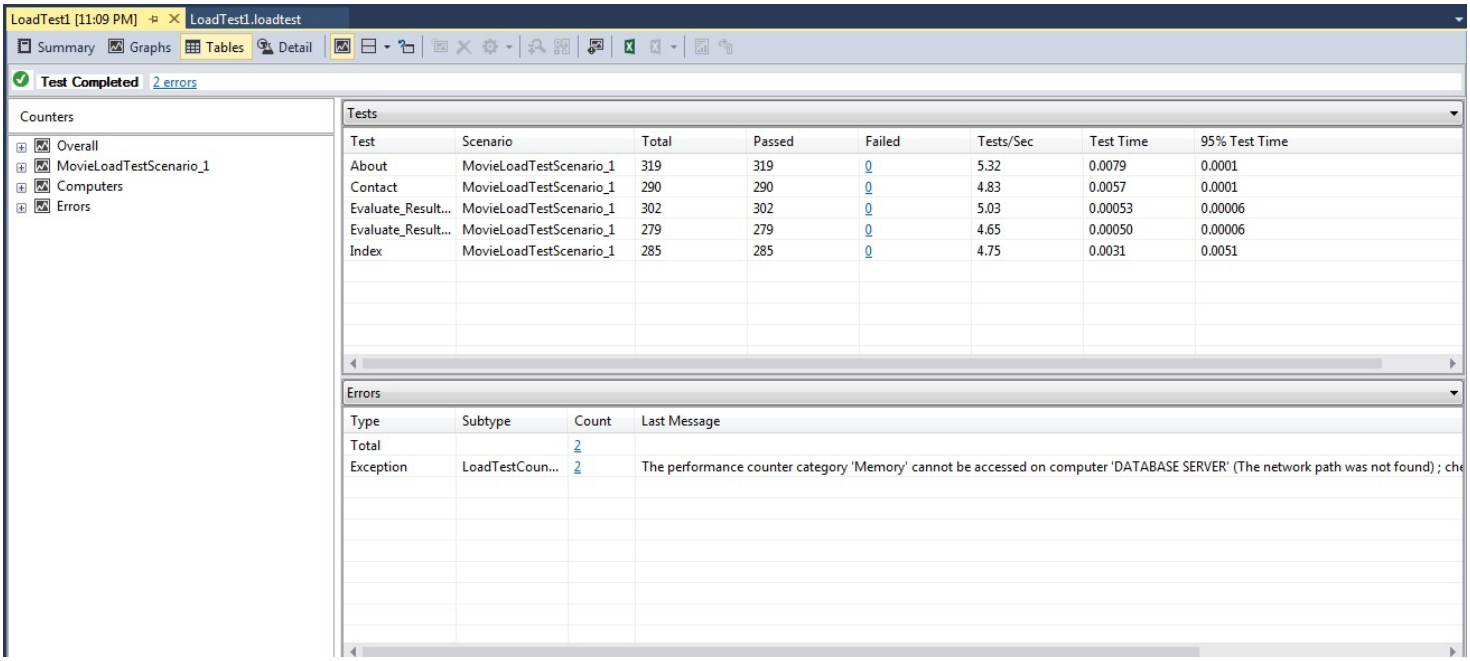
So, now we have everything in place, I'll just go ahead and click the button on the top left and then it will produce the below result.



Now, when you click on Graphs, it will display the how specific test scenarios behaved.



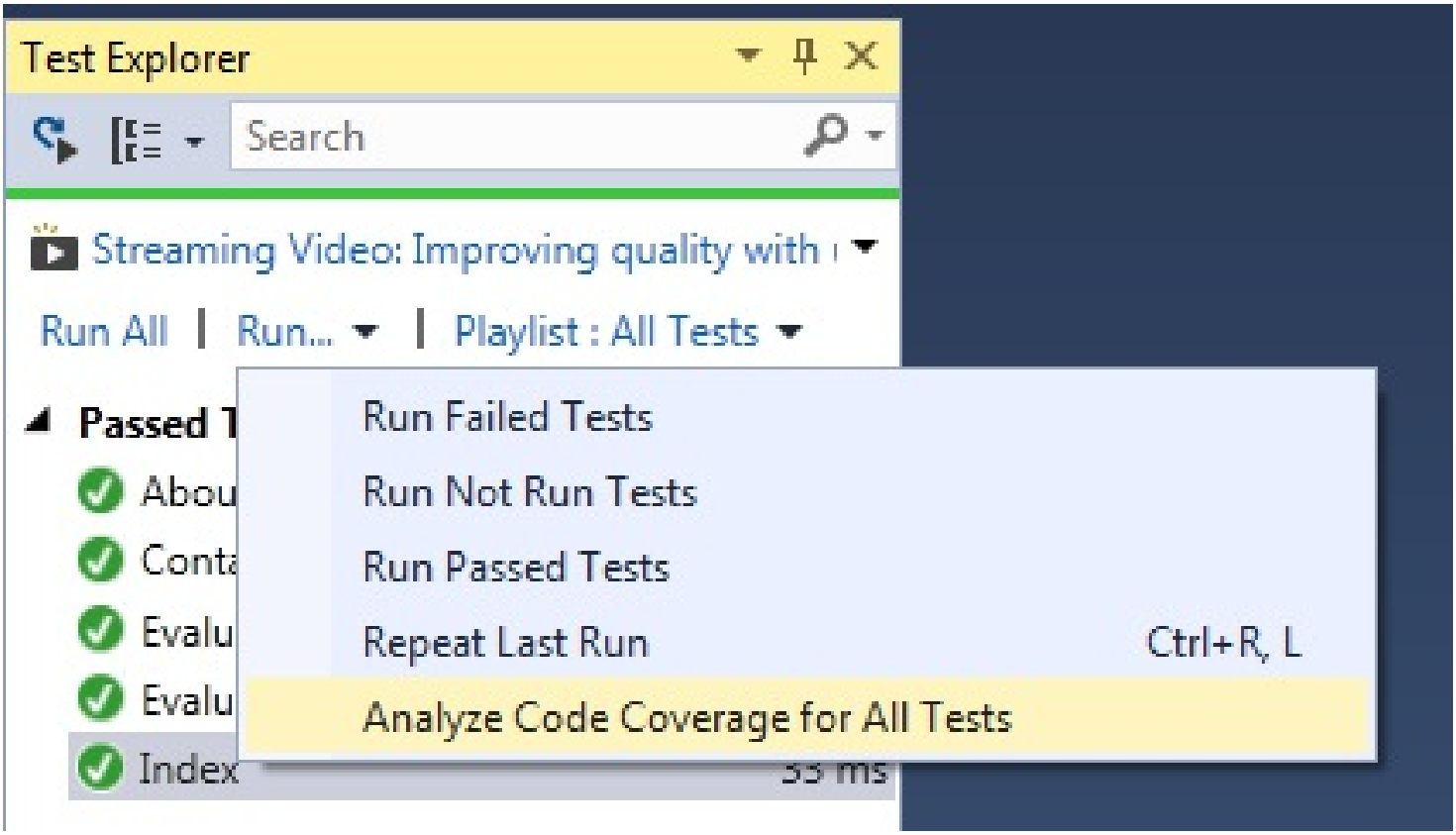
You can also analyze the snapshot of the test counters which you have added like shown below

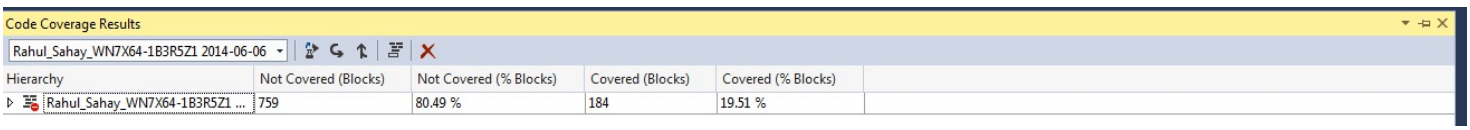


Code Coverage:-

Code Coverage is one of the best things which give you the snapshot of Test Cases. Means, the test cases which you have written, how much code it is

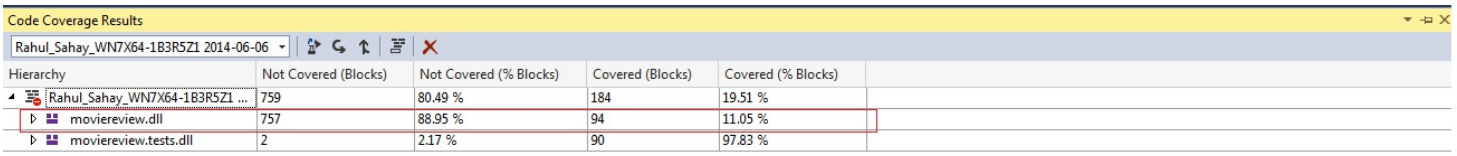
actually covering in your application. So, when you run the Code Coverage from the Test Explorer after running all the tests, it will give you statistics as shown below.





So, as you can see from the statistics, it

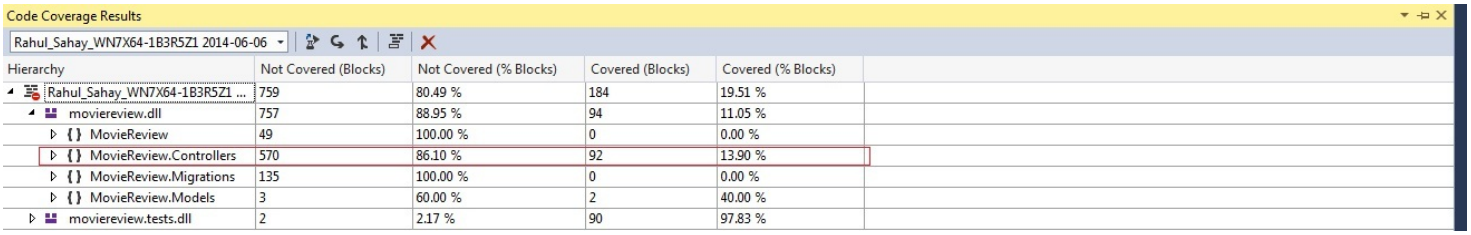
is giving not covered percentage as 80.49% which is very huge. Reason being, I have just written test case only for Home Controller Index action. So, now, when I open this and go inside it I can see my results and that is quite satisfactory percentage.



Code Coverage Results

Rahul_Sahay_WN7X64-1B3R5Z1 2014-06-06

Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
Rahul_Sahay_WN7X64-1B3R5Z1 ...	759	80.49 %	184	19.51 %
moviereview.dll	757	88.95 %	94	11.05 %
moviereview.tests.dll	2	2.17 %	90	97.83 %

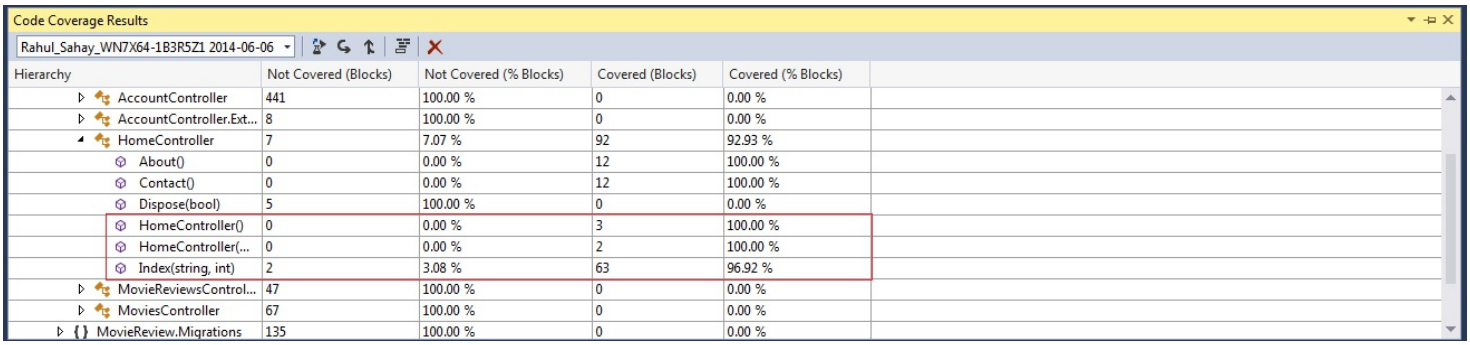


Code Coverage Results

Rahul_Sahay_WN7X64-1B3R5Z1 2014-06-06

Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
Rahul_Sahay_WN7X64-1B3R5Z1 ...	759	80.49 %	184	19.51 %
moviereview.dll	757	88.95 %	94	11.05 %
{ } MovieReview	49	100.00 %	0	0.00 %
{ } MovieReview.Controllers	570	86.10 %	92	13.90 %
{ } MovieReview.Migrations	135	100.00 %	0	0.00 %
{ } MovieReview.Models	3	60.00 %	2	40.00 %
moviereview.tests.dll	2	2.17 %	90	97.83 %

Now, when I open this controller's section I can see home controller in there is 96% which is a very good percentage.



Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
AccountController	441	100.00 %	0	0.00 %
AccountController.Ext...	8	100.00 %	0	0.00 %
HomeController	7	7.07 %	92	92.93 %
About()	0	0.00 %	12	100.00 %
Contact()	0	0.00 %	12	100.00 %
Dispose(bool)	5	100.00 %	0	0.00 %
HomeController()	0	0.00 %	3	100.00 %
HomeController(...	0	0.00 %	2	100.00 %
Index(string, int)	2	3.08 %	63	96.92 %
MovieReviewsControl...	47	100.00 %	0	0.00 %
MoviesController	67	100.00 %	0	0.00 %
{ } MovieReview.Migrations	135	100.00 %	0	0.00 %

Summary:-

So, in this chapter, we have covered one of the most important things of MVC that is Unit Testing. So, we have seen how to test different logics of our application by simply writing different test case also by writing some of the helper classes or associated methods. We have also seen how to test UI with Web Tests and Load tests. Apart from testing, we have also covered code coverage, which basically gives the snapshot of your test cases.

Now, I would like you guys to make this habit always writing test cases for your application.

Chapter 10: Deployment

WHAT DO you find in this CHAPTER?

⇒ Introduction

⇒ Configuration Files

⇒ Hosting ASP.Net MVC Application

⇒ Deployment Preparation

⇒ Deployment on IIS

⇒ Deployment on Azure

⇒ Summary

Introduction:-

Hi friends, I hope you guys have enjoyed building an ASP.Net MVC application End to End. Now, we have done with our Development and Testing. So, now it's time for Deployment. Now, in this

entire module, I'll discuss various techniques involved for deployment and also the minor details which you must follow while deployment. I'll start the discussion with different types of configuration files involved in ASP.Net application, and then will go through the key ingredients required for hosting any ASP.Net MVC Application. Once, that is done then I'll 1st host the site on IIS and then online on Azure.

Configuration Files:-

So, basically configuration files in .NET are nerve of ASP.Net architecture. They control everything about execution

environment. So, settings like

- **Authentication**
- **Compilation**
- **Custom Errors**
- **Cryptography**
- **Connections**

are basically controlled by configuration files. So, the settings which I have listed above are not the only ones which are controlled by configuration files. It controls many more things. The configuration setting in .NET is pretty much extendable. So, you can write your own custom configuration. Hence, what you need is appropriate classes to support them. Now, when I say my custom setting, I can go ahead and define

my custom setting inside the section "***<appSettings>***". Like below in the sample, I have added one Key with one value and I'll retrieve the same in my web application using one special class known as "***ConfigurationManager***".

```
<appSettings>
  <add key="webpages:Version" value="2.0.0.0" />
  <add key="webpages:Enabled" value="false" />
  <add key="PreserveLoginUrl" value="true" />
  <add key="ClientValidationEnabled" value="true" />
  <add key="UnobtrusiveJavaScriptEnabled" value="true" />
  <add key="elmah.mvc.disableHandler" value="false" />
  <add key="elmah.mvc.disableHandleErrorFilter" value="false" />
  <add key="elmah.mvc.requiresAuthentication" value="true" />
  <add key="elmah.mvc.IgnoreDefaultRoute" value="false" />
  <add key="elmah.mvc.allowedRoles" value="Admin" />
  <add key="elmah.mvc.allowedUsers" value="*" />
  <add key="elmah.mvc.route" value="elmah" />
  <add key="MovieTest" value="Testing"/>
</appSettings>
```

Now, let me retrieve the same in my view.

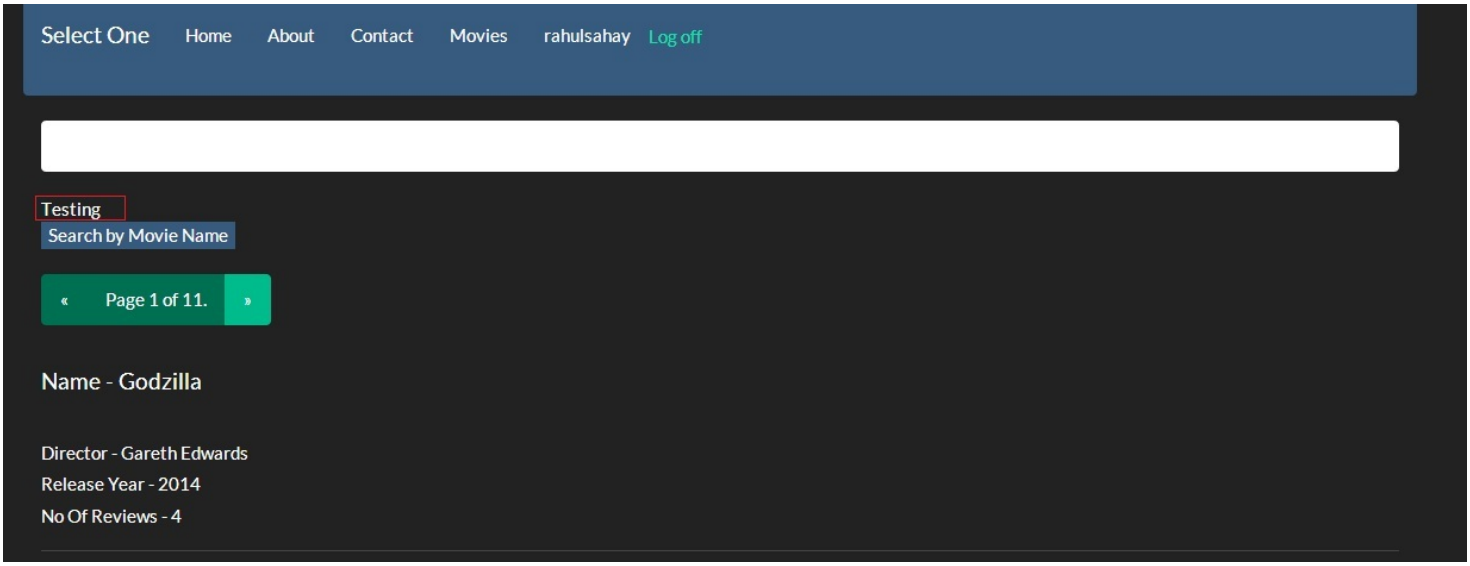
Index.cshtml | Web.config

```
1  @using System.Configuration
2  @model IPagedList<MovieReview.Models.MovieViewModel>
3  @{
4      ViewBag.Title = "Home Page";
5  }
6
7  <div>
8      <form method="get" action="@Url.Action("Index")" data-movie-ajax="true" data-movie-target="#movieDiv">
9          <input type="search" class="form-control has-success" name="SearchQuery">
10         <br />
11         @ConfigurationManager.AppSettings["MovieTest"]
12         <br />
13         <input type="submit" class="btn-primary" value="Search by Movie Name" />
14     </form>
15 </div>
16
17 @*@using (Ajax.BeginForm(
18     new AjaxOptions
19     {
20         HttpMethod = "get",
21         InsertionMode = InsertionMode.Replace,
22         UpdateTargetId = "movieDiv"
23     }))
24 {
25     <input type="search" class="form-control has-success" name="SearchQuery">
26     <br />
27     <input type="submit" class="btn-primary" value="Search by Movie Name" />
28 }*@
29
30
31 @Html.Partial("_movies",Model)
```

Namespace for ConfigurationManager Class.

Retrieving custom setting value.

Now, with the above code in place, I could see my custom setting value on the home page.



So, basically "***ConfigurationManager***" is the basic class which can read "***Appsetting value***" by just providing the key. Now, the value which I have retrieved directly in view, you can grab the same value from the controller itself. What you can do you can just put this setting in the "***viewbag***" and then in the view you can just output the result. So, basically config files are a placeholder which holds settings value means which you don't want to hard code for your application.

One important thing to understand here, that configuration files in .NET here are hierarchical means the configuration in your application is not only governed by

the "***web.config***" file which you are having in the root of your project but also dependent on "***configuration files***" at higher level. So, basically for any MVC application configuration path follows route

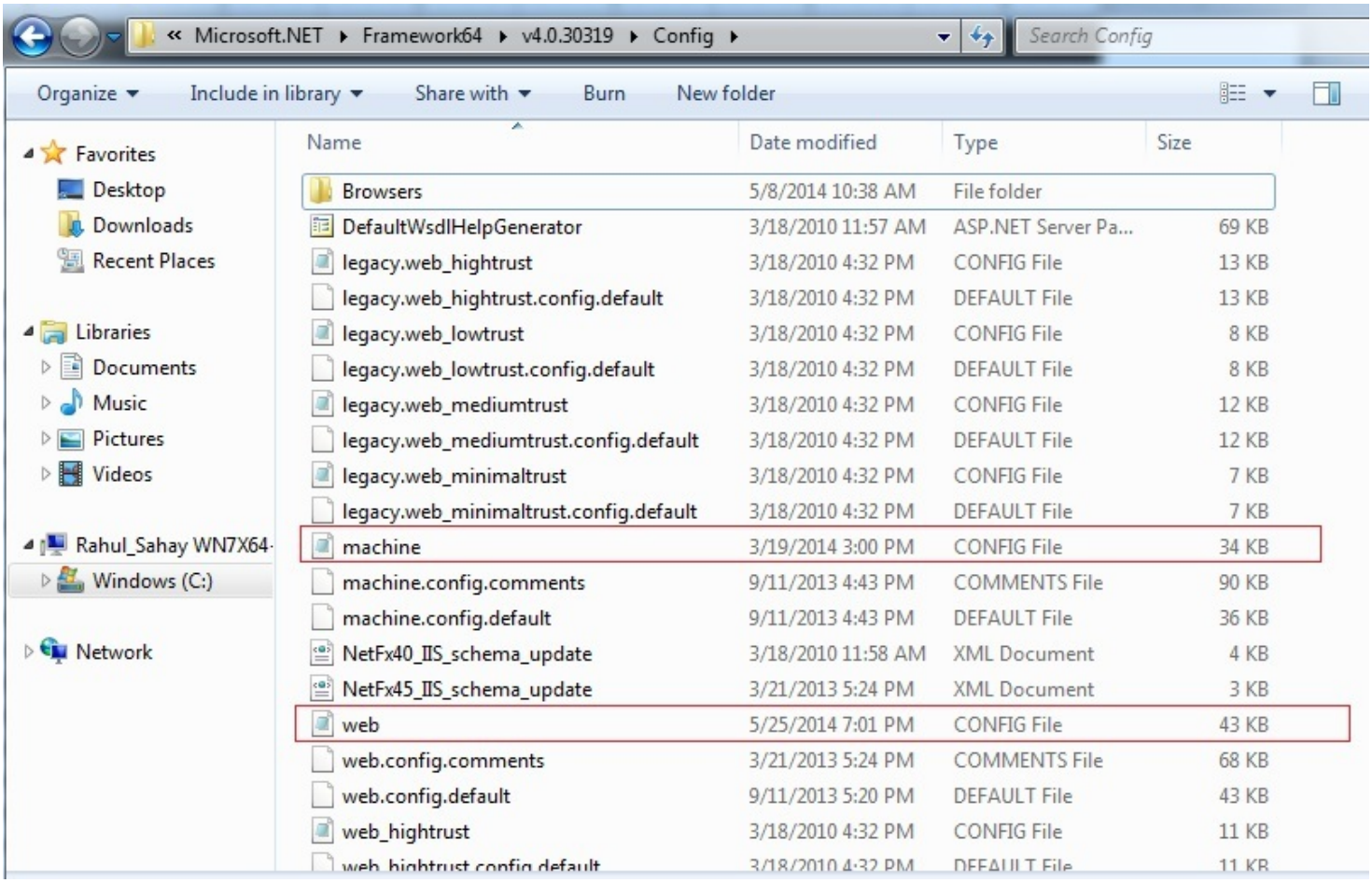
- **machine.config**
- **machine Web.config**
- **Parent Web.config**
- **Your project web.config**

So, "***machine.config***" file controls all the basic settings for your .Net applications that run on your machine. Now, all .NET applications irrespective of its nature are governed by this Machine.config file. Now, there is also machine level web.config file; we have

already looked at this file when we have discussed **Health Monitoring** in the previous chapter. So, basically it puts all the default settings for every **ASP.NET Web Application** running on the machine. Now, the next thing could be or couldn't be your parent web.config. It actually means the scenario where in your project is dependent on some other primary project. So, basically your project web.config file at that time is going to inherit the settings from the parent's project web.config file.

However, inside my project web.config file, I can go ahead and override machine level config or machine level web.config file settings. So, if you open

the machine config file by navigating the path
"***C:\Windows\Microsoft.NET\Framew***
You will actual see both the files in there as highlighted below in the screen shot.



Now, when you open machine.config file, you can see some of the ASP.Net

settings as shown below.

```
<httpModules>
    <add name="OutputCache" type="System.Web.Caching.OutputCacheModule" />
    <add name="Session" type="System.Web.SessionState.SessionStateModule" />
    <add name="WindowsAuthentication" type="System.Web.Security.WindowsAuthenticationModule" />
    <add name="FormsAuthentication" type="System.Web.Security.FormsAuthenticationModule" />
    <add name="PassportAuthentication" type="System.Web.Security.PassportAuthenticationModule" />
    <add name="RoleManager" type="System.Web.Security.RoleManagerModule" />
    <add name="UrlAuthorization" type="System.Web.Security.UrlAuthorizationModule" />
    <add name="FileAuthorization" type="System.Web.Security.FileAuthorizationModule" />
    <add name="AnonymousIdentification" type="System.Web.Security.AnonymousIdentificationModule" />
    <add name="Profile" type="System.Web.Profile.ProfileModule" />
    <add name="ErrorHandlerModule" type="System.Web.Mobile.ErrorHandlerModule, System.Web.Mobile, Version=4.0.0.0, Cultu
    <add name="ServiceModel" type="System.ServiceModel.Activation.HttpModule, System.ServiceModel.Activation, Version=4.
    <add name="UrlRoutingModule-4.0" type="System.Web.Routing.UrlRoutingModule" />
    <add name="ScriptModule-4.0" type="System.Web.Handlers.ScriptModule, System.Web.Extensions, Version=4.0.0.0, Culture
</httpModules>
```

Like this, we have many more settings in this file which govern ASP.NET application setting in your application. Similarly, we also have web.config file at machine level which also governs default settings in our application. Now, apart from these web.config files; in our application we have one more web.config file inside the views folder which governs the razor settings as shown below.

Web.config | Index.cshtml | Web.config

```
<?xml version="1.0"?>

<configuration>
  <configSections>
    <sectionGroup name="system.web.webPages.razor" type="System.Web.WebPages.Razor.Configuration.RazorWebSectionGroup, System.We
      <section name="host" type="System.Web.WebPages.Razor.Configuration.HostSection, System.Web.WebPages.Razor, Version=2.0.0.0
      <section name="pages" type="System.Web.WebPages.Razor.Configuration.RazorPagesSection, System.Web.WebPages.Razor, Version=
    </sectionGroup>
  </configSections>

  <system.web.webPages.razor>
    <host factoryType="System.Web.Mvc.MvcWebRazorHostFactory, System.Web.Mvc, Version=4.0.0.0, Culture=neutral, PublicKeyToken=3
    <pages pageBaseType="System.Web.Mvc.WebViewPage">
      <namespaces>
        <add namespace="System.Web.Mvc" />
        <add namespace="System.Web.Mvc.Ajax" />
        <add namespace="System.Web.Mvc.Html" />
        <add namespace="System.Web.Optimization"/>
        <add namespace="System.Web.Routing" />
        <add namespace="PagedList"/>
        <add namespace="PagedList.Mvc"/>
      </namespaces>
    </pages>
  </system.web.webPages.razor>

  <appSettings>
    <add key="webpages:Enabled" value="false" />
  </appSettings>

  <system.web>
    <httpHandlers>
      <add path="*" verb="*" type="System.Web.HttpNotFoundHandler"/>
    </httpHandlers>
```

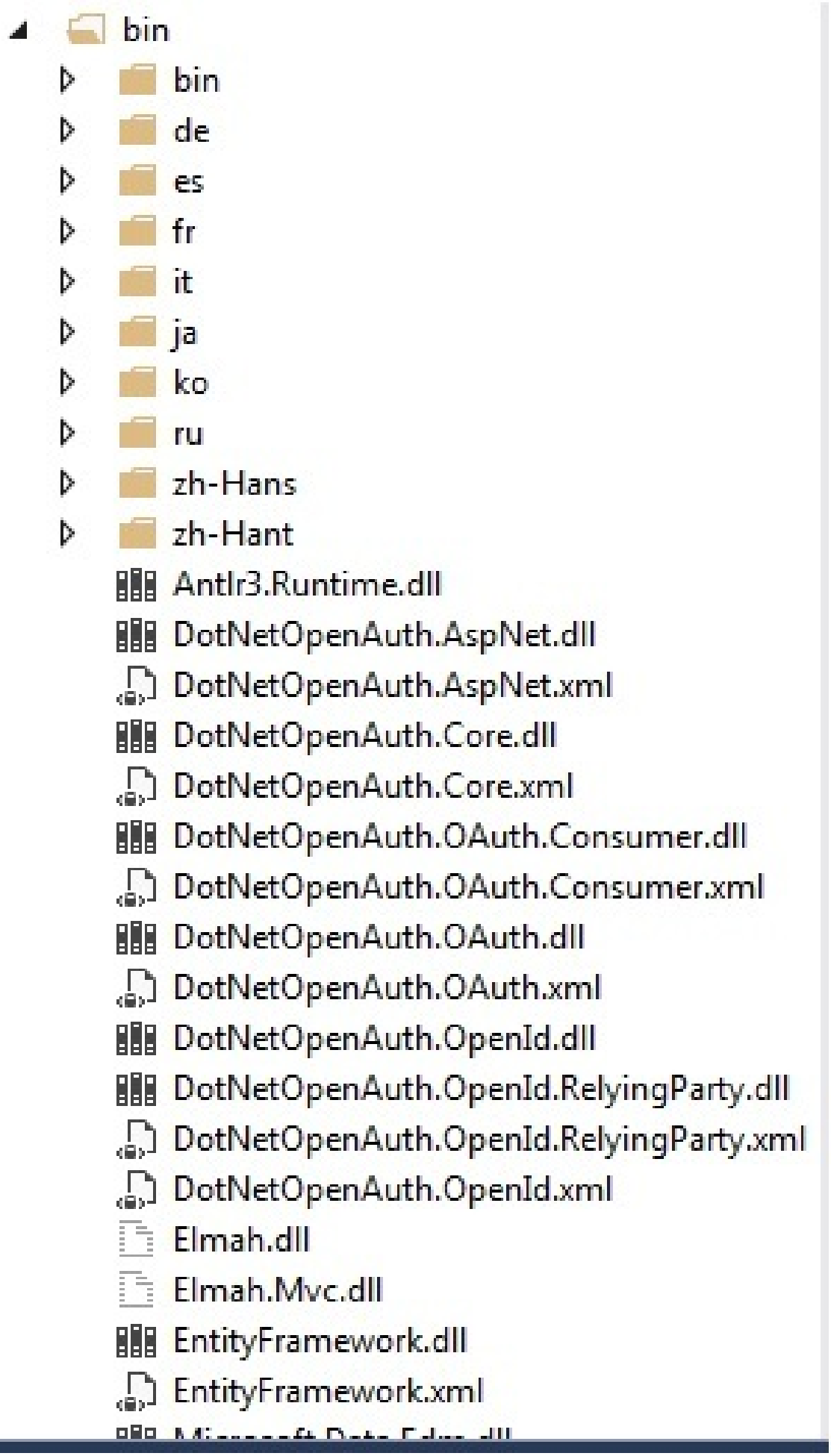
So, as you can see above we have added couple of namespaces for paged List earlier. So, these namespaces are basically required to provide you the intellisense in the views. Now, another interesting setting that is mentioned here is "***<httpHandlers>***". So, here what this component is doing if any request arrives which is not resolved by your

MVC application, then it will be picked by "***HttpNotFoundHandler***". Now, this component is responsible for returning 404 errors to the client. This is essentially here to ensure that any user cannot directly go to the view like ***http://localhost:1034/home/index.cshtr*** One must follow the routing rule; hence request must come from the controller directly and it's controller's responsibility to choose a view and redirect user on the page.

Hosting ASP.Net MVC Application:-

Now, since we have understood the

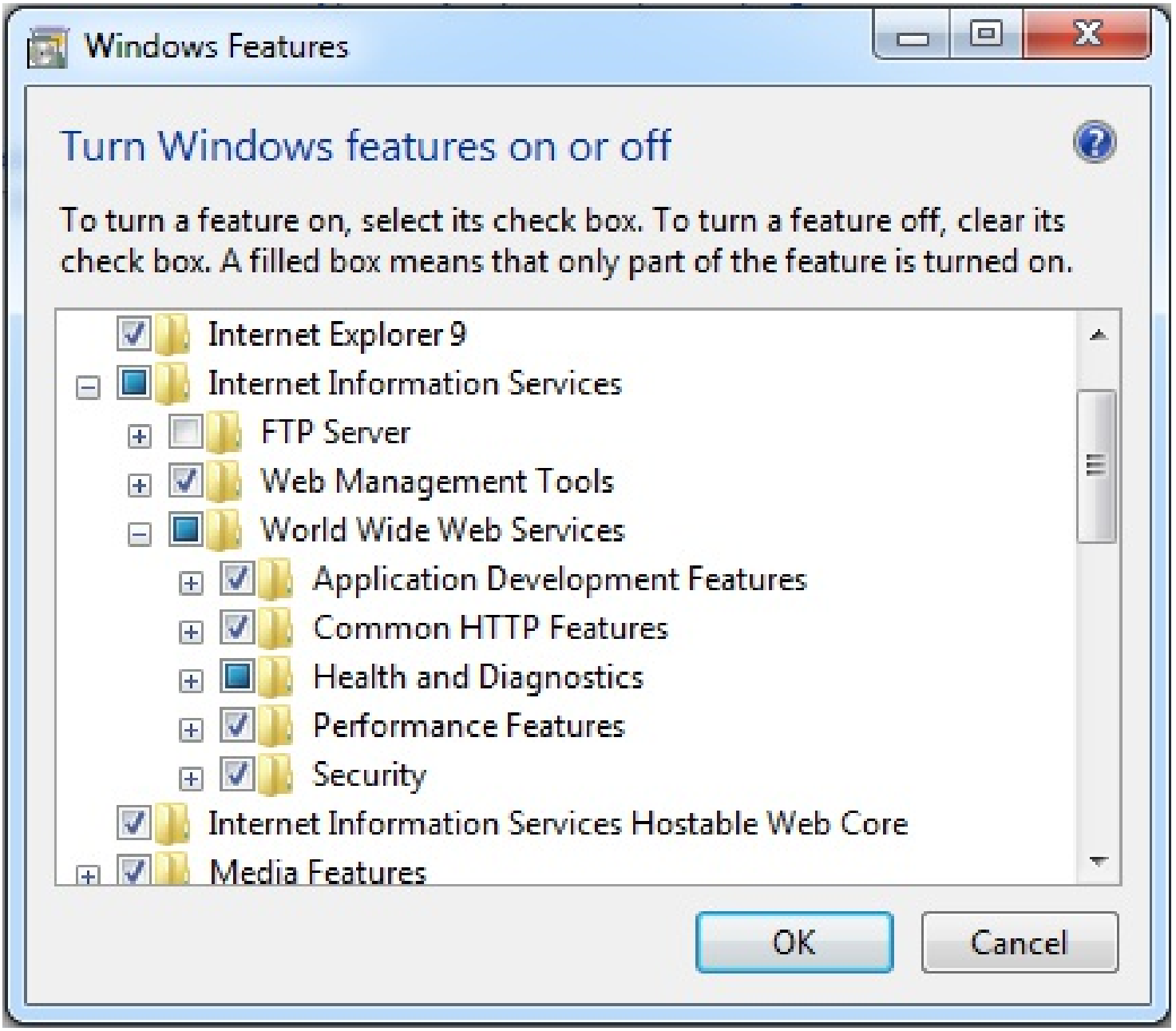
basics of ASP.Net configuration. So, it's time to move on and start the process for deployment. So, basically whenever we are building any MVC application, we are basically producing a "***dll***" file associated for the same with all the application settings and logic inside. Now, these "***dlls***" sit in the bin directory of the project as shown below in the screen shot.


bin
bin
de
es
fr
it
ja
ko
ru
zh-Hans
zh-Hant
Antlr3.Runtime.dll
DotNetOpenAuth.AspNet.dll
DotNetOpenAuth.AspNet.xml
DotNetOpenAuth.Core.dll
DotNetOpenAuth.Core.xml
DotNetOpenAuth.OAuth.Consumer.dll
DotNetOpenAuth.OAuth.Consumer.xml
DotNetOpenAuth.OAuth.dll
DotNetOpenAuth.OAuth.xml
DotNetOpenAuth.OpenId.dll
DotNetOpenAuth.OpenId.RelyingParty.dll
DotNetOpenAuth.OpenId.RelyingParty.xml
DotNetOpenAuth.OpenId.xml
Elmah.dll
Elmah.Mvc.dll
EntityFramework.dll
EntityFramework.xml

Now, one important point to understand here **dlls** are not **self hostable**. They need some kind of mechanism to get hosted. And for websites the host process is **WebServer**, where in we configure the application and IIS settings accordingly. Now, this **HOST** process also takes care of delivering HTTP requests inside our application **dll**. So far during the course of development, we have been using "***IIS Express***". Now, IIS Express makes the development process very easy and very smooth, it takes our development setting what we are building and produce the output.

Now, one important thing is **IIS Express**

runs with our **Identity** and we can any time **Stop** or **Start** the **WebServer** whenever we want. But, when we are deploying the application on **Internet** or **Intranet**, then in that case we'll be using complete version of **IIS**. So, we can install IIS on any version of windows. But, it is off by default on our machines. So, for enabling the same we can go into "***Windows Features***" from the control panel and check the required setting as shown below in the screen shot.



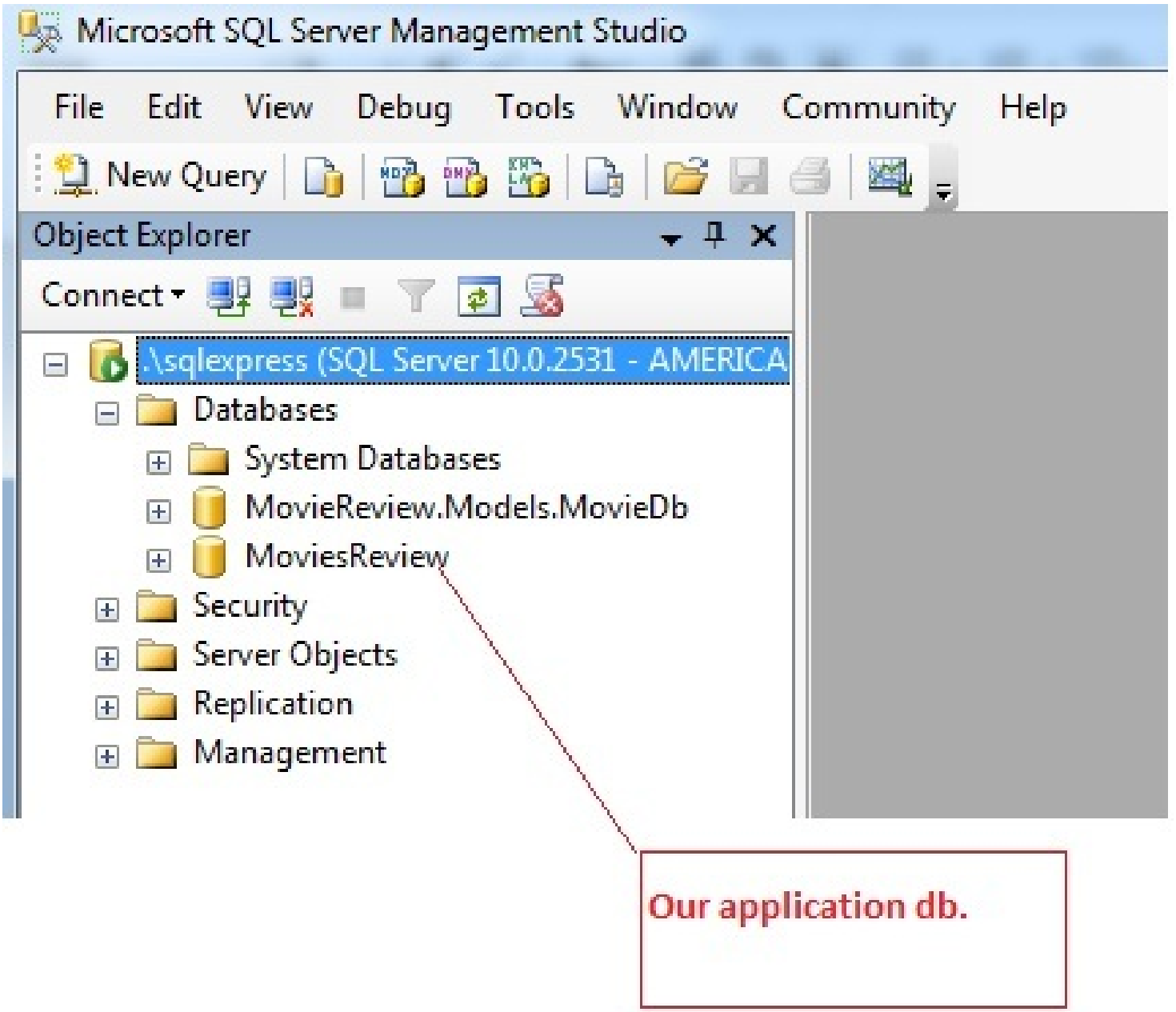
So, as you can see in the screen shot, I have already installed IIS and its associated component on my machine. Now, one more option you can consider for installing all these things via "***Web***

Platform Installer". Also, I have already installed **SQL Server Express Edition**. Now, after installing IIS, you can verify the same from the browser as shown below in the screen shot. You just need to navigate to "***localhost***" and it will produce the default page for the IIS.

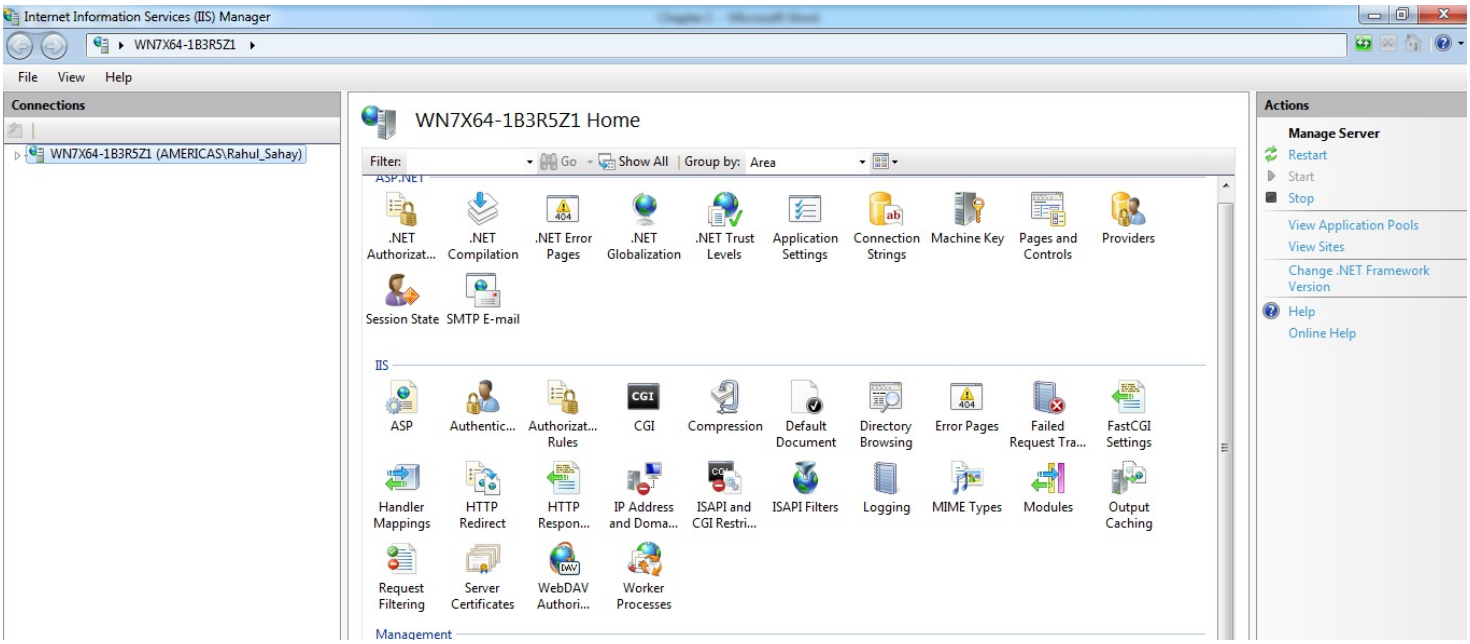


Also, my SQL instance is running fine. You can also verify the same by logging

into the same as shown below:-



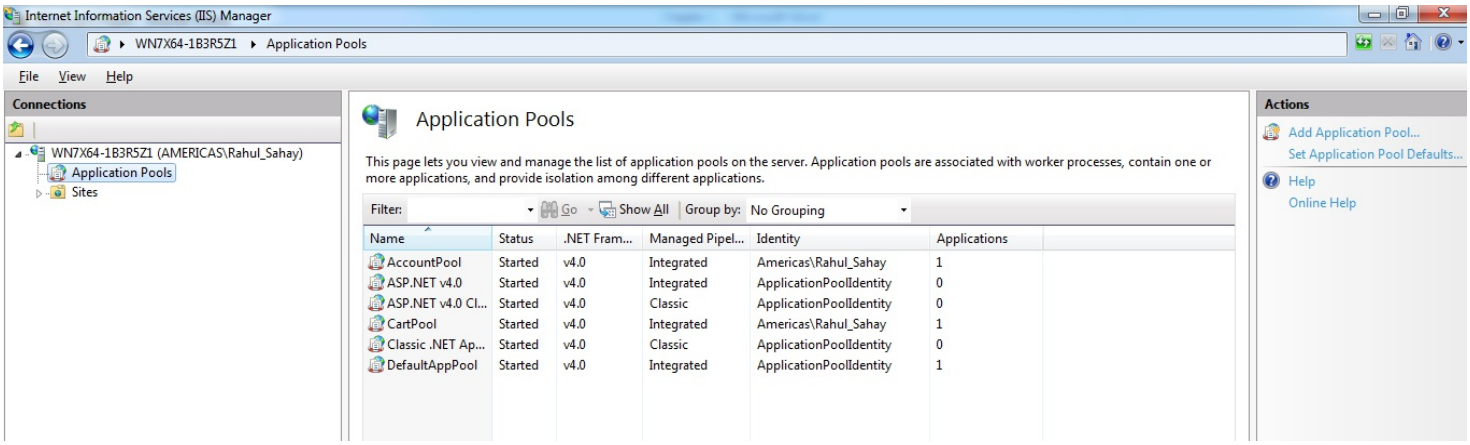
Now, below is the dashboard for our IIS Settings.



Deployment Preparation:-

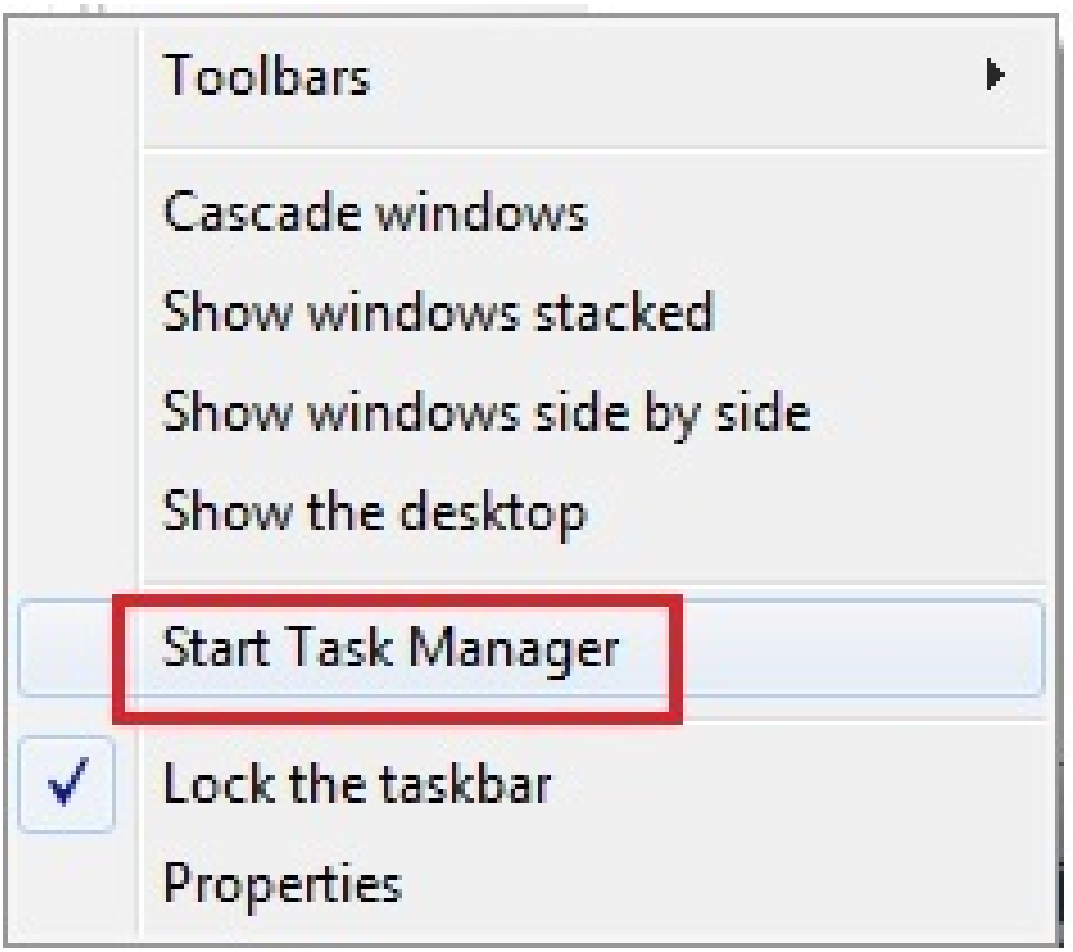
So, as shown above there are tons of IIS settings listed in this dashboard which you can go ahead configure for your application. Now, all these settings which you will be setting or changing will be mapped to xml files or config files which we have been looking earlier in this chapter. Now, when I expand the

machine node I can see two sub nodes, one with the Application Pool and the other which is having Sites. Now, an **Application Pool** is a process in which a web application will run.



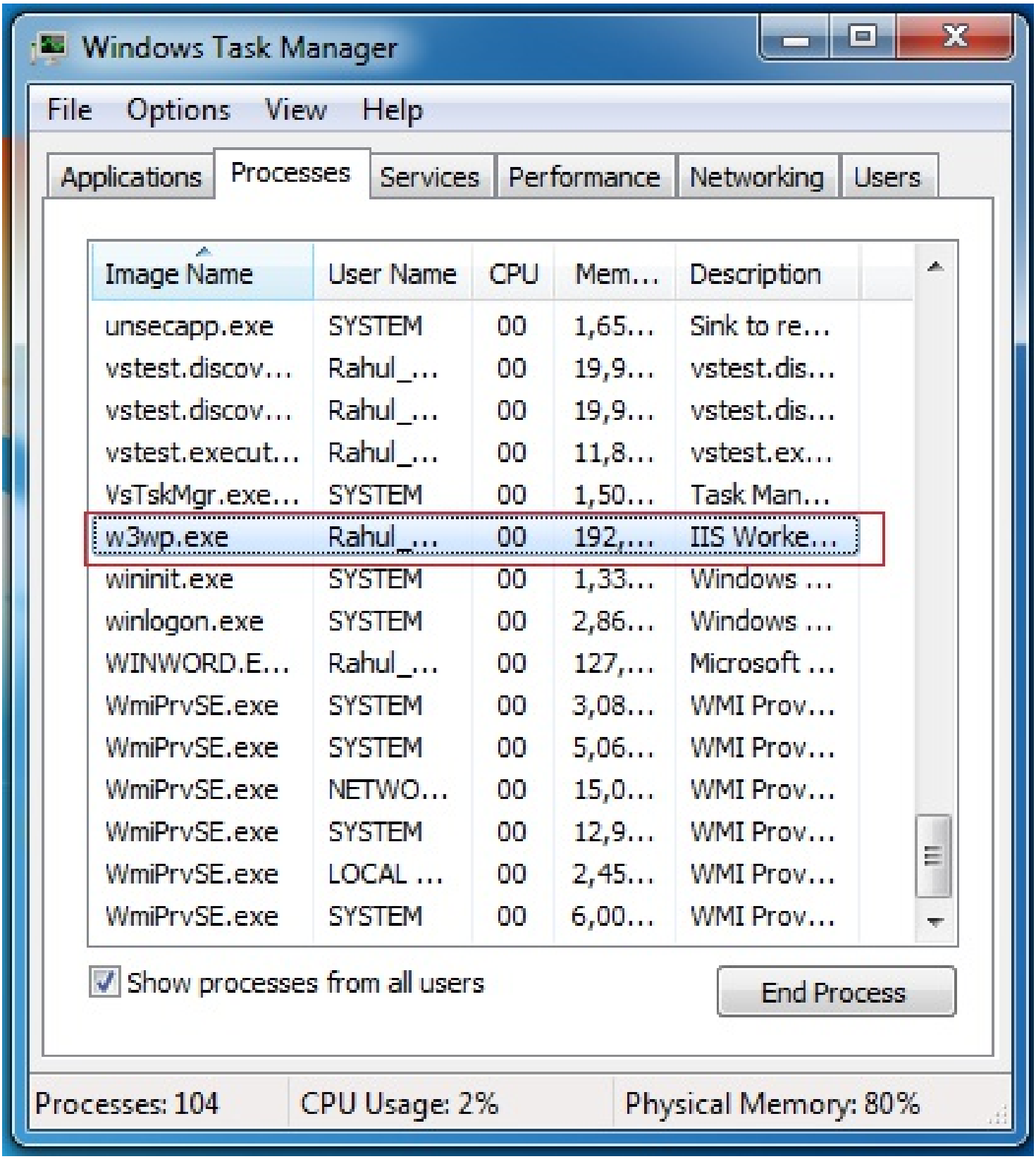
Now, here we can go ahead and create our custom pools as well. But, here I'll be using "***DefaultAppPool***" the last one listed in this list. Now, as you can see that there is already one application running in there. And this is the default IIS Application, which we have just

seen, when we navigated to ***http://localhost/***. Now, if we go to task manager of windows by just right clicking on the task bar and selecting "***Start Task Manger***".



And, when you open the task manager, then you will find one process with the

name "***w3wp.exe***" as shown below. This is actually "***World wide web worker process***" exe. So, this will be the process which will host our MVC Application.



Now, there are certain tasks which I

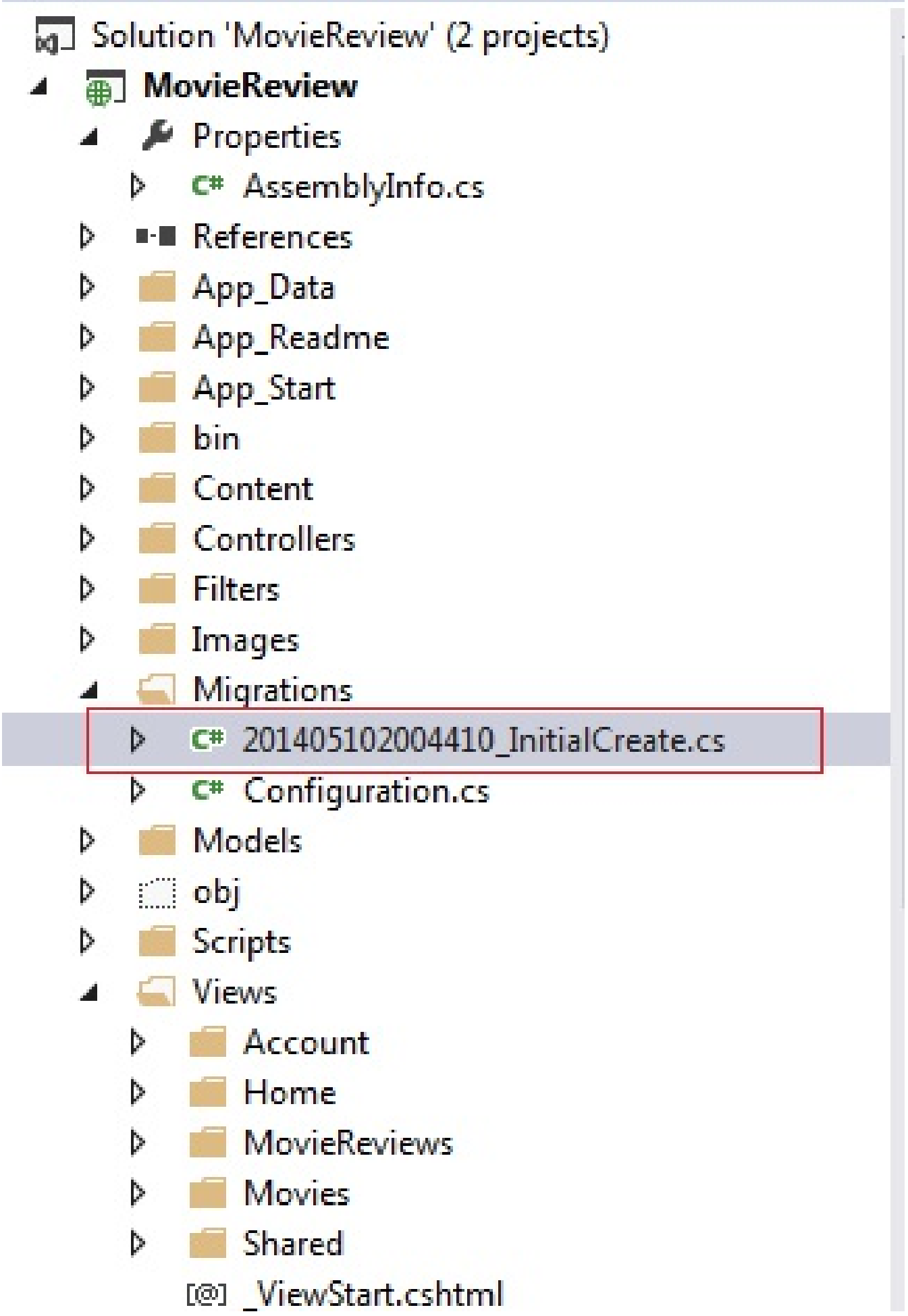
need to configure before deployment. So, the 1st thing I'll do is change our migration setting. Now, I'll make "***AutomaticMigrationsEnabeld***" set to false. We have set this to true during development just to make sure that our models are in sync.

```
public Configuration()
        {
            AutomaticMigrationsEnabled = false;
        }
```

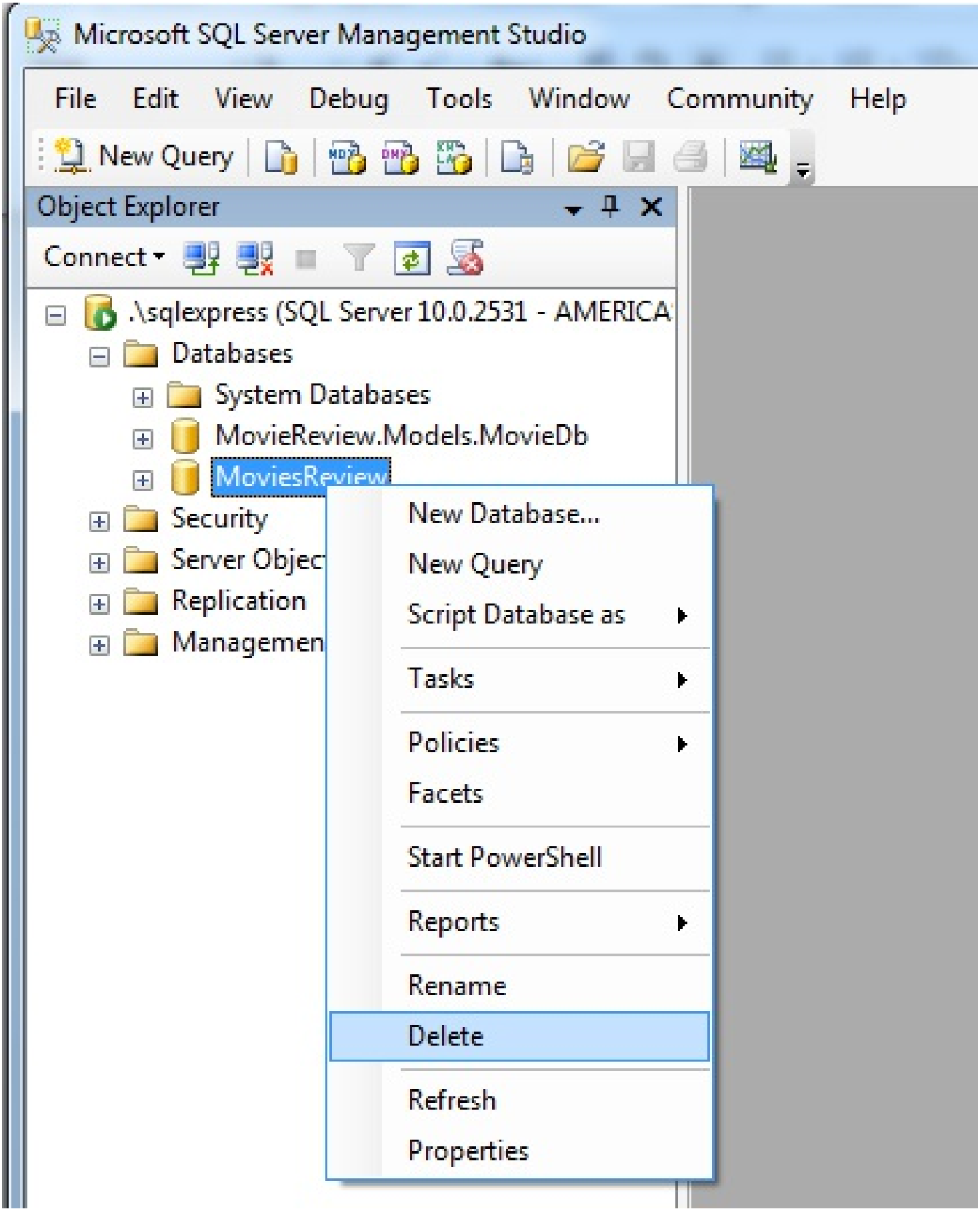
Now, the next thing which I am going to do is restart the migration process. So, now I also want to comment out the section from where I injected 100 movies to test sorting and paging.

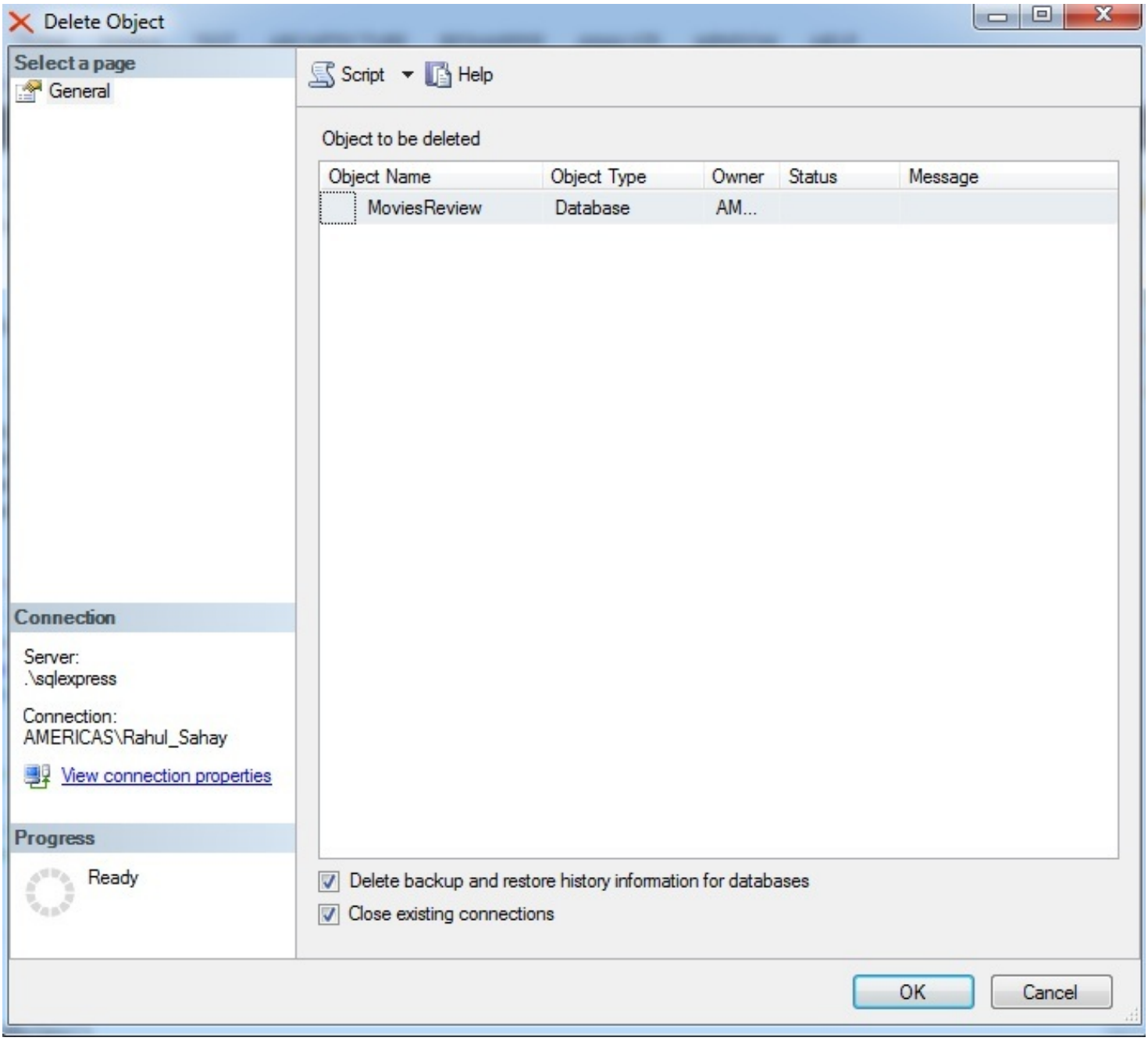
```
  //for (int i = 0; i < 100; i++)
            //{
            //    context.Movies.AddOrUpdate(m => m.MovieName,
            //        new Movie { MovieName = "Anony_Movie_" + i.ToString(),
DirectorName = "Anony_Director_" + i.ToString(), ReleaseYear = "2014" });
            //}
```

Then, I'll also delete my initial create database script as highlighted below.


Solution 'MovieReview' (2 projects)
MovieReview
Properties
AssemblyInfo.cs
References
App_Data
App_Readme
App_Start
bin
Content
Controllers
Filters
Images
Migrations
201405102004410_InitialCreate.cs
Configuration.cs
Models
obj
Scripts
Views
Account
Home
MovieReviews
Movies
Shared
_ViewStart.cshtml

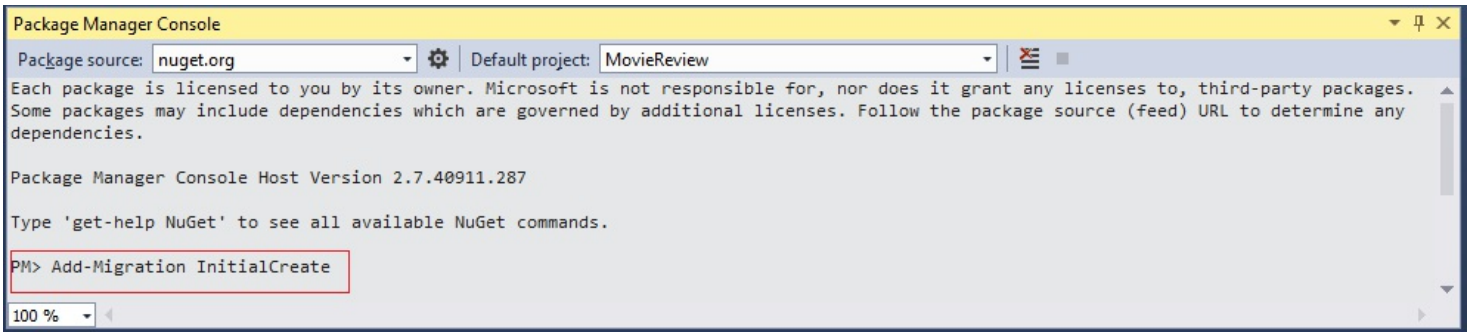
Now, I'll also create the same from scratch using a new database and the definition of schema will be based on what is in our model right now. Now, the best way to do the same with Entity Framework itself. But, before this I'll go ahead and delete our existing database as shown below from the management studio.

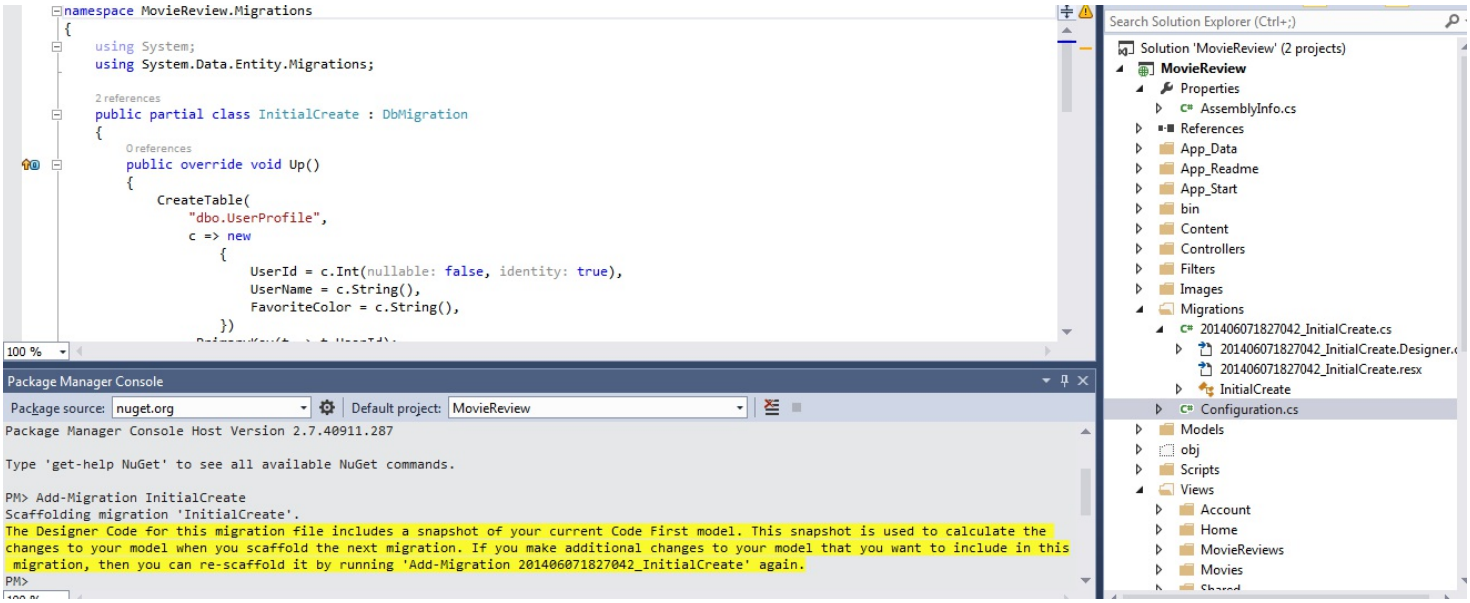

Microsoft SQL Server Management Studio
File
Edit
View
Debug
Tools
Window
Community
Help
New Query
Object Explorer
Connect
.\sqlexpress (SQL Server 10.0.2531 - AMERICA
Databases
System Databases
MovieReview.Models.MovieDb
MoviesReview
Security
Server Objec
Replication
Managemen
New Database...
New Query
Script Database as
Tasks
Policies
Facets
Start PowerShell
Reports
Rename
Delete
Refresh
Properties



So, now we have successfully dropped our database. Now, let's open the package manager console and let's create our initial migration file with the PowerShell command “Add-Migration

InitialCreate". Also, since we are not using "***Automatic Migartions***", so we won't be able to go ahead and say "***Update-Database***" from the powershell command.





So, in the above screen shot, we can see

that command executed successfully and hence created the brand new "***InitialCreate***" file. Now, below is the script which is generated after running the command.

```
namespace MovieReview.Migrations
```

```
namespace MovieReview.Migrations
{
    using System;
    using System.Data.Entity.Migrations;

    public partial class InitialCreate : DbMigration
    {
        public override void Up()
        {
            CreateTable(
                "dbo.UserProfile",
                c => new
                    {
                        UserId = c.Int(nullable: false, identity: true),
                        UserName = c.String(),
                        FavoriteColor = c.String(),
                    })
                .PrimaryKey(t => t.UserId);

            CreateTable(
                "dbo.Movies",
                c => new
                    {
                        Id = c.Int(nullable: false, identity: true),
                        MovieName = c.String(),
```

```
                    DirectorName = c.String(),
                    ReleaseYear = c.String(),
                })
            .PrimaryKey(t => t.Id);

        CreateTable(
            "dbo.MovieReviews",
            c => new
                {
                    Id = c.Int(nullable: false, identity: true),
                    ReviewerName = c.String(),
                    ReviewerComments = c.String(nullable: false, maxLength:
200),
                    ReviewerRating = c.Int(nullable: false),
                    MovieId = c.Int(nullable: false),
                })
            .PrimaryKey(t => t.Id)
            .ForeignKey("dbo.Movies", t => t.MovieId, cascadeDelete: true)
            .Index(t => t.MovieId);

    }

    public override void Down()
    {
        DropIndex("dbo.MovieReviews", new[] { "MovieId" });
        DropForeignKey("dbo.MovieReviews", "MovieId", "dbo.Movies");
        DropTable("dbo.MovieReviews");
        DropTable("dbo.Movies");
        DropTable("dbo.UserProfile");
    }
}
}
```

So, here you can see that it includes everything which we have been using in our application. Now, the next thing which I want to do here is to ensure that

when it gets deployed, it should run the migration automatically. But, before that I want to put a checkpoint where it will make sure that "***WebSecurity***" doesn't get initialized more than once. Otherwise, it will throw exception here and everything will get stop here itself.

```
  if (!WebSecurity.Initialized)
            {
                WebSecurity.InitializeDatabaseConnection("DefaultConnection",
"UserProfile", "UserId", "UserName", autoCreateTables: true);
            }
```

Now, I also need to put the same checkpoint in the "***global.asax***" file as well as shown below in the screen shot.

```
using System.Web;
using System.Web.Http;
using System.Web.Mvc;
using System.Web.Optimization;
using System.Web.Routing;
using WebMatrix.WebData;

namespace MovieReview
{
    // Note: For instructions on enabling IIS6 or IIS7 classic mode,
    // visit http://go.microsoft.com/?LinkId=9394801

    0 references
    public class MvcApplication : System.Web.HttpApplication
    {
        0 references
        protected void Application_Start()
        {
            if (!WebSecurity.Initialized)
            {
                WebSecurity.InitializeDatabaseConnection("DefaultConnection", "UserProfile", "UserId", "UserName", autoCreateTab
            }
            AreaRegistration.RegisterAllAreas();

            WebApiConfig.Register(GlobalConfiguration.Configuration);
            FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            BundleConfig.RegisterBundles(BundleTable.Bundles);
            AuthConfig.RegisterAuth();
        }
    }
}
```

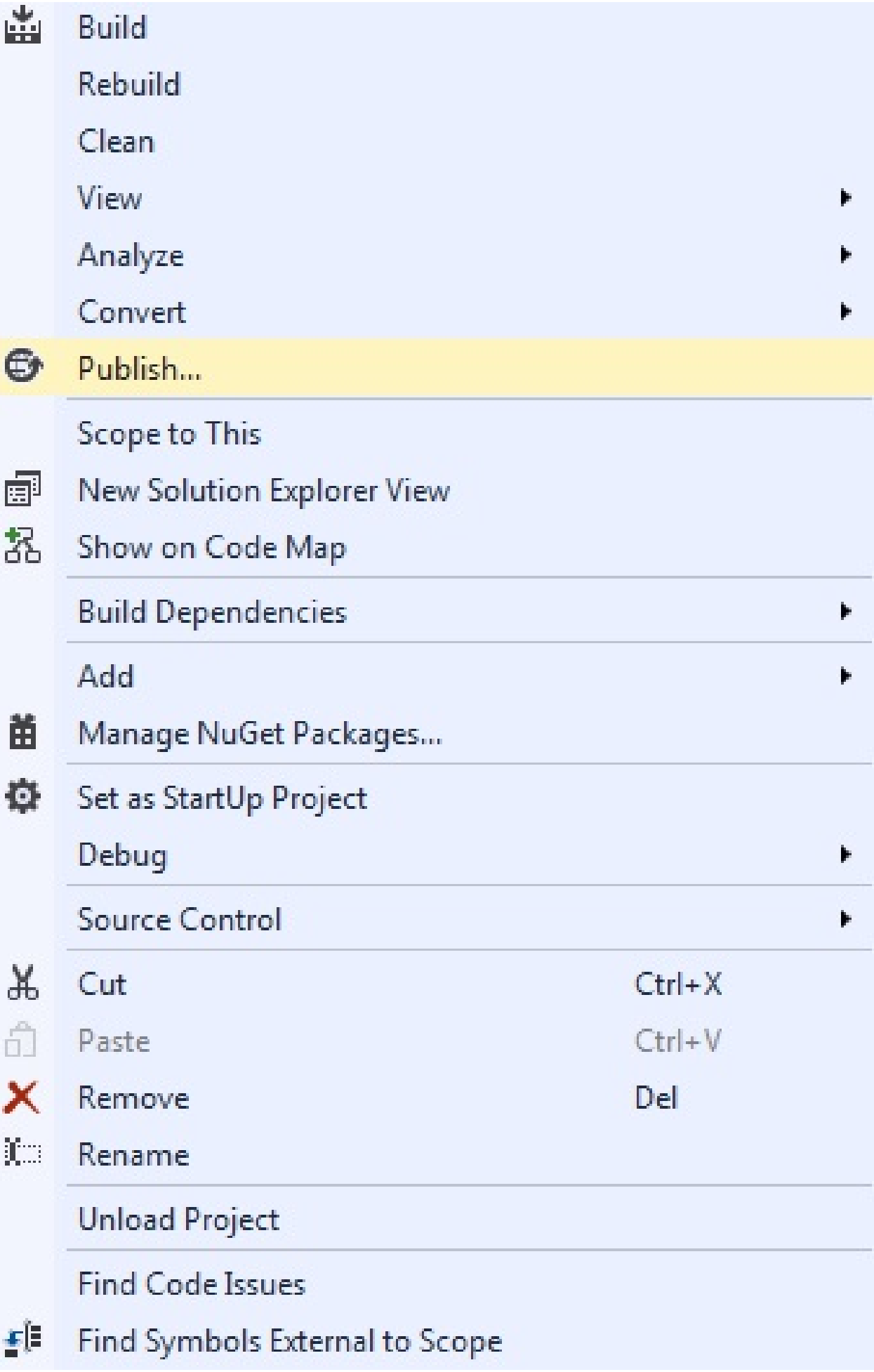
Now, in the "***application start***" event itself I would like run the migrations, now I can do the same using "***DbMigrator***" as shown below

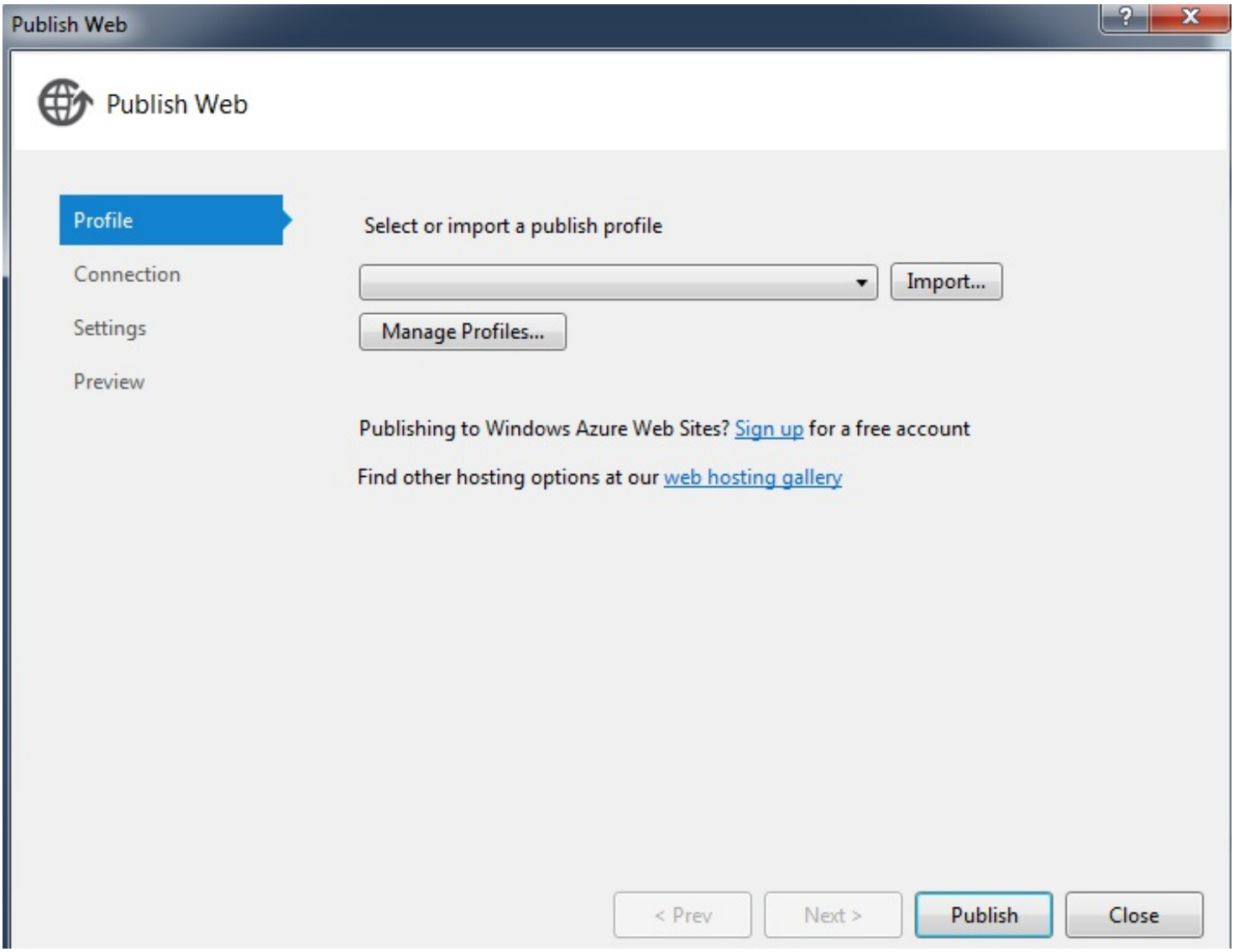
```
var migrator = new DbMigrator(new Configuration());
migrator.Update();
```

Now, what this will do, it will run the seed method and make sure that database is populated properly.

Deployment on IIS:-

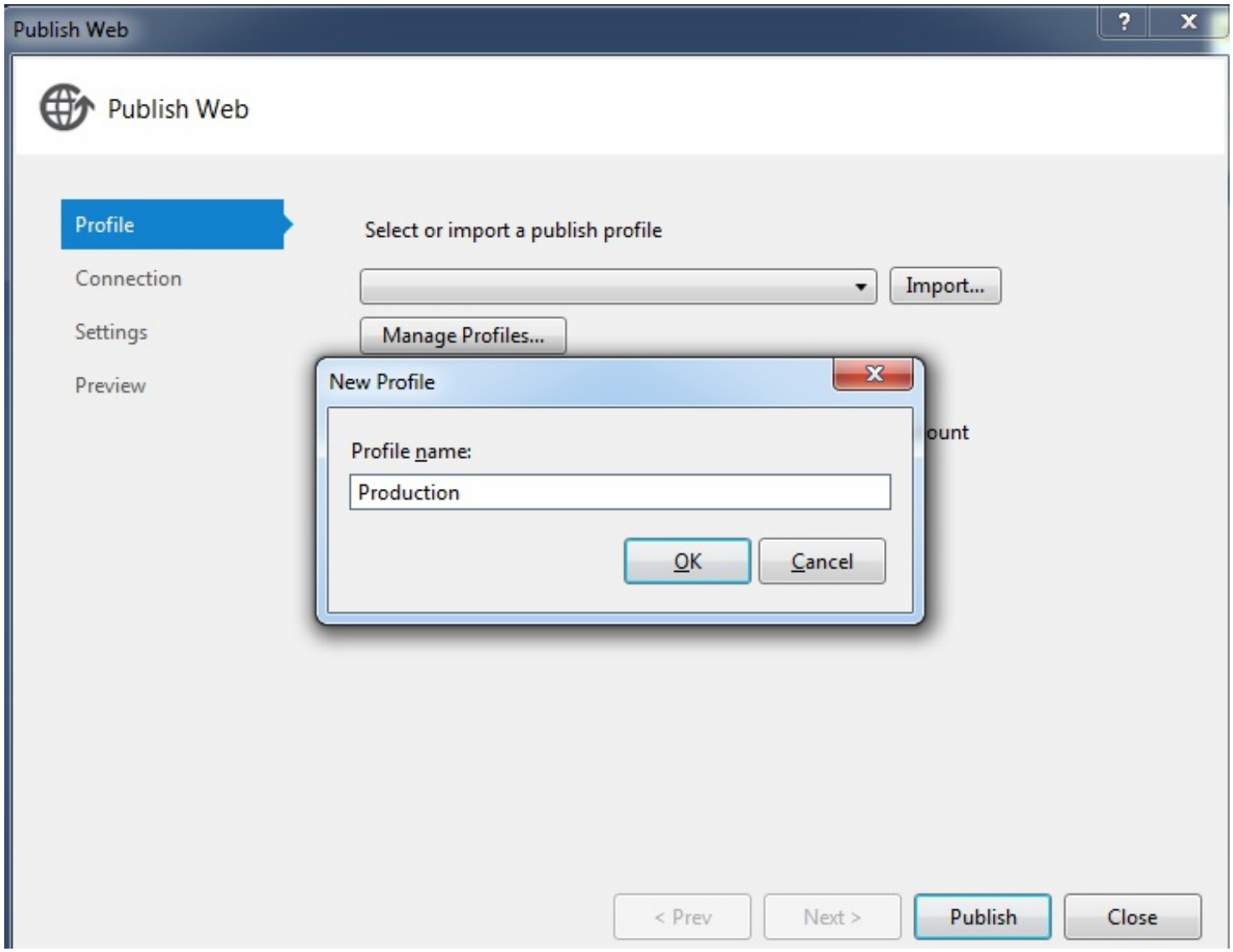
Now, I think we are good to go with IIS Deployment. So, when you are ready to deploy the project, what you can do, right-click the project and click on publish and this will produce the below options.


Build
Rebuild
Clean
View
Analyze
Convert
Publish...
Scope to This
New Solution Explorer View
Show on Code Map
Build Dependencies
Add
Manage NuGet Packages...
Set as StartUp Project
Debug
Source Control
Cut Ctrl+X
Paste Ctrl+V
Remove Del
Rename
Unload Project
Find Code Issues
Find Symbols External to Scope



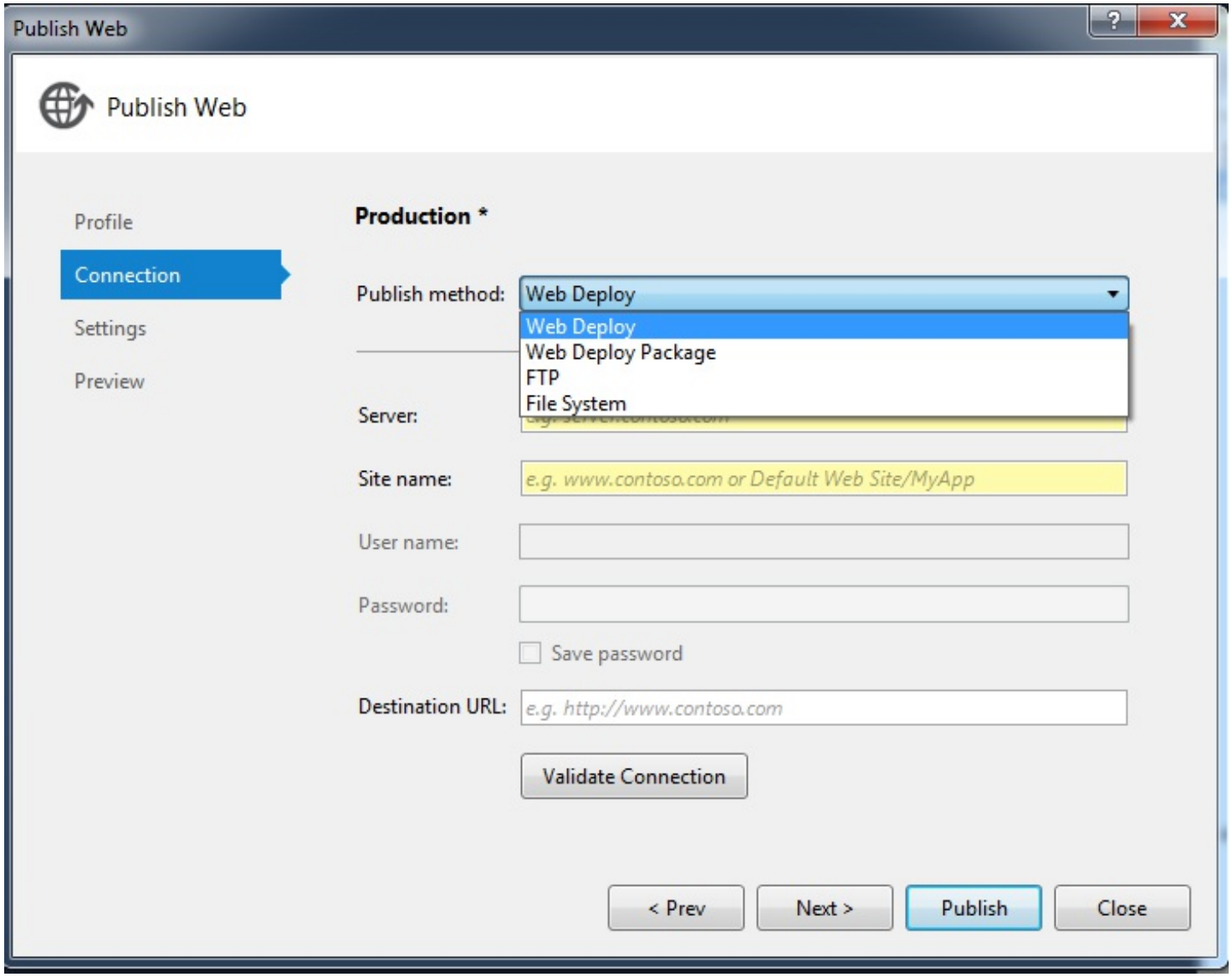
So, Visual Studio will figure out all the things like what are the files that need to deployed on the web server, collect them and push them there. So, the 1st thing I need to do here is create new

profile. For doing that I will click on the dropdown and click on New Profile by giving there some profile name. Let's say "***Production***" as profile name.



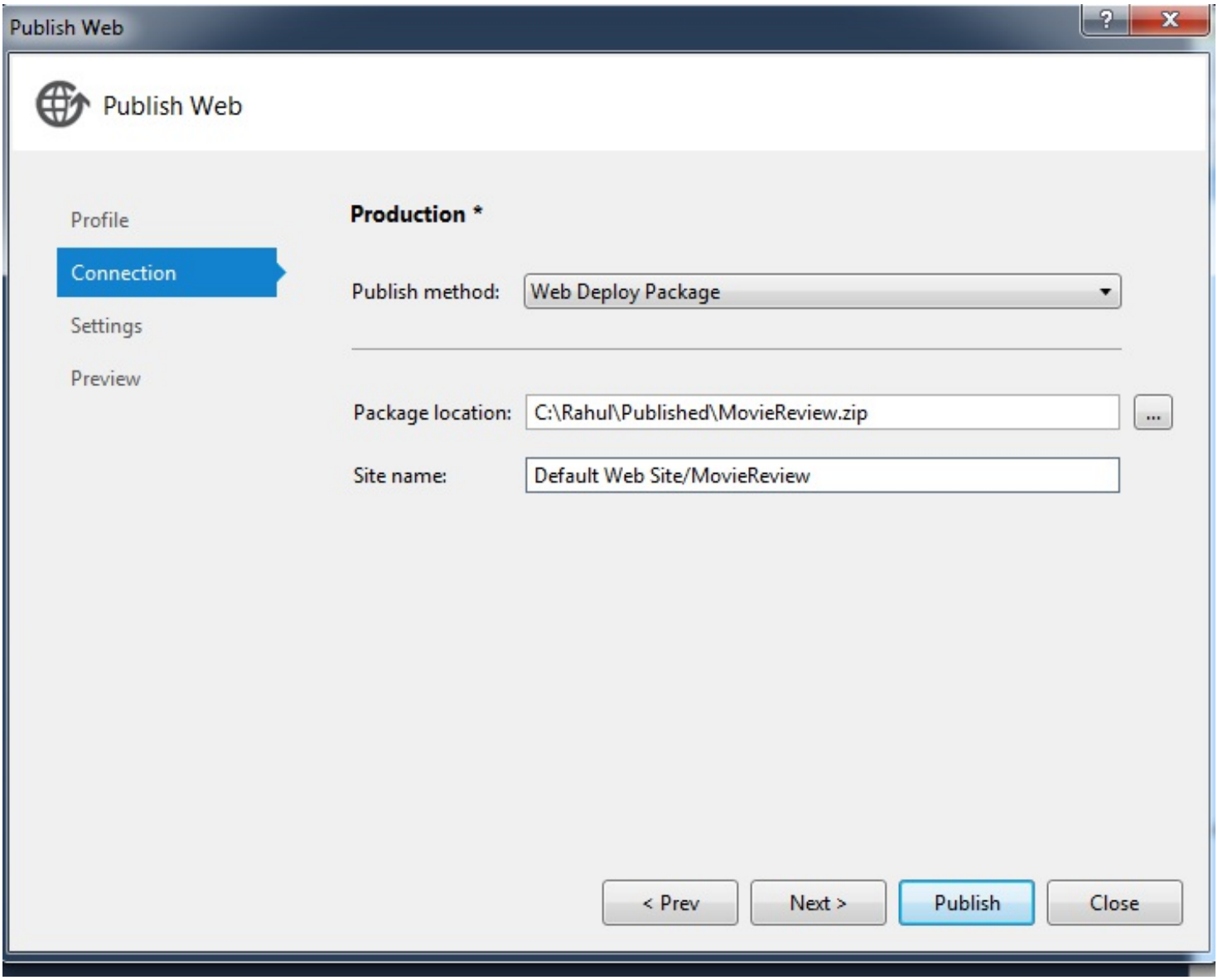
And then on the next screen I will select

the publish method, now here you will get multiple options like shown below



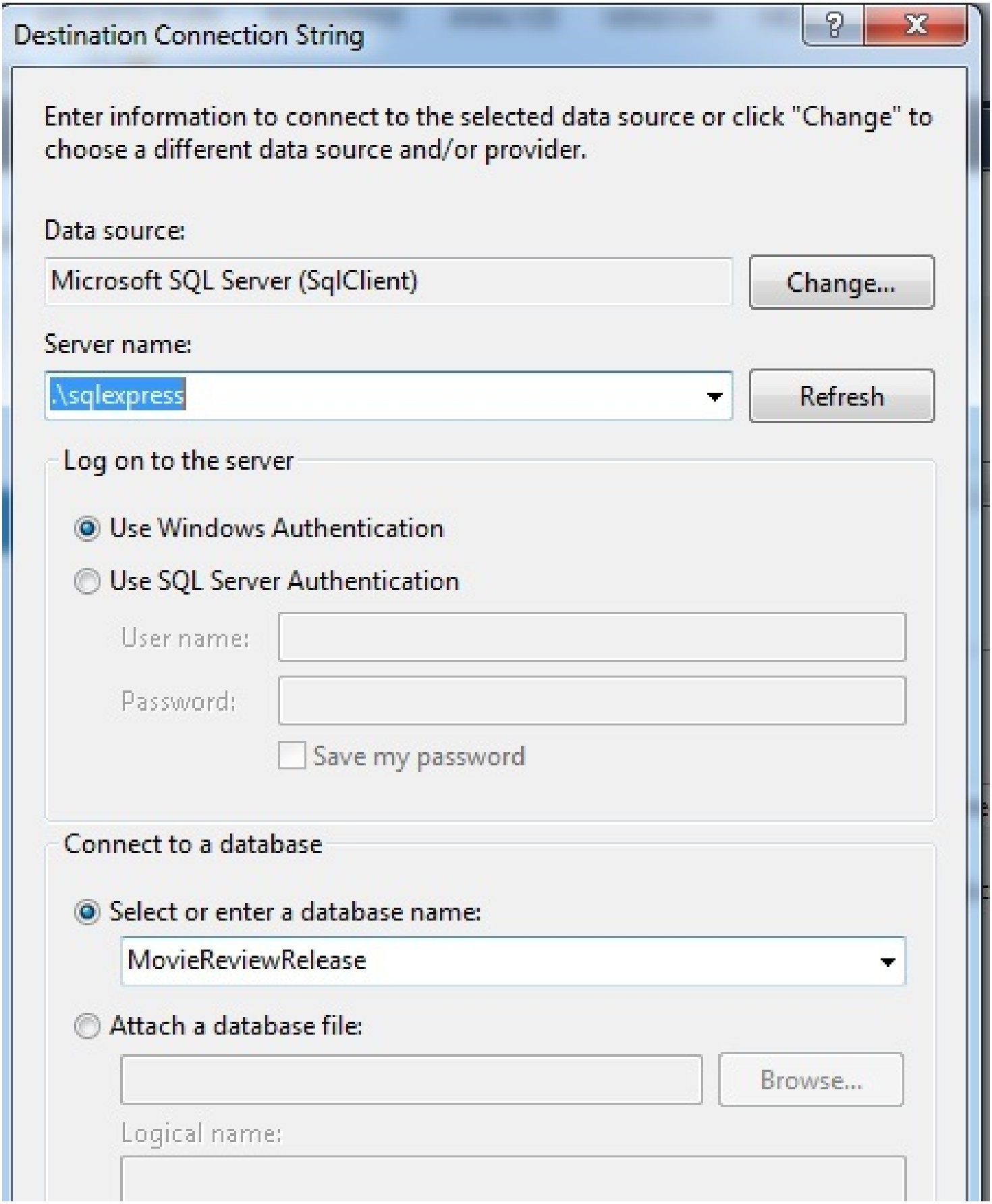
So, if you would like to FTP, then Visual Studio will do that for you. You can also push the files on the file system. Now,

you can also use Web Deploy option, then you need to specify the server name and site name as well, then it will directly push the files on that server. So, the option here what I am going to use is web deploy package. Now, a web deploy package is a zip file; you can simply take this zip file and install the same on the webserver. Now, I'll provide the below parameters as shown below in the screen shot.

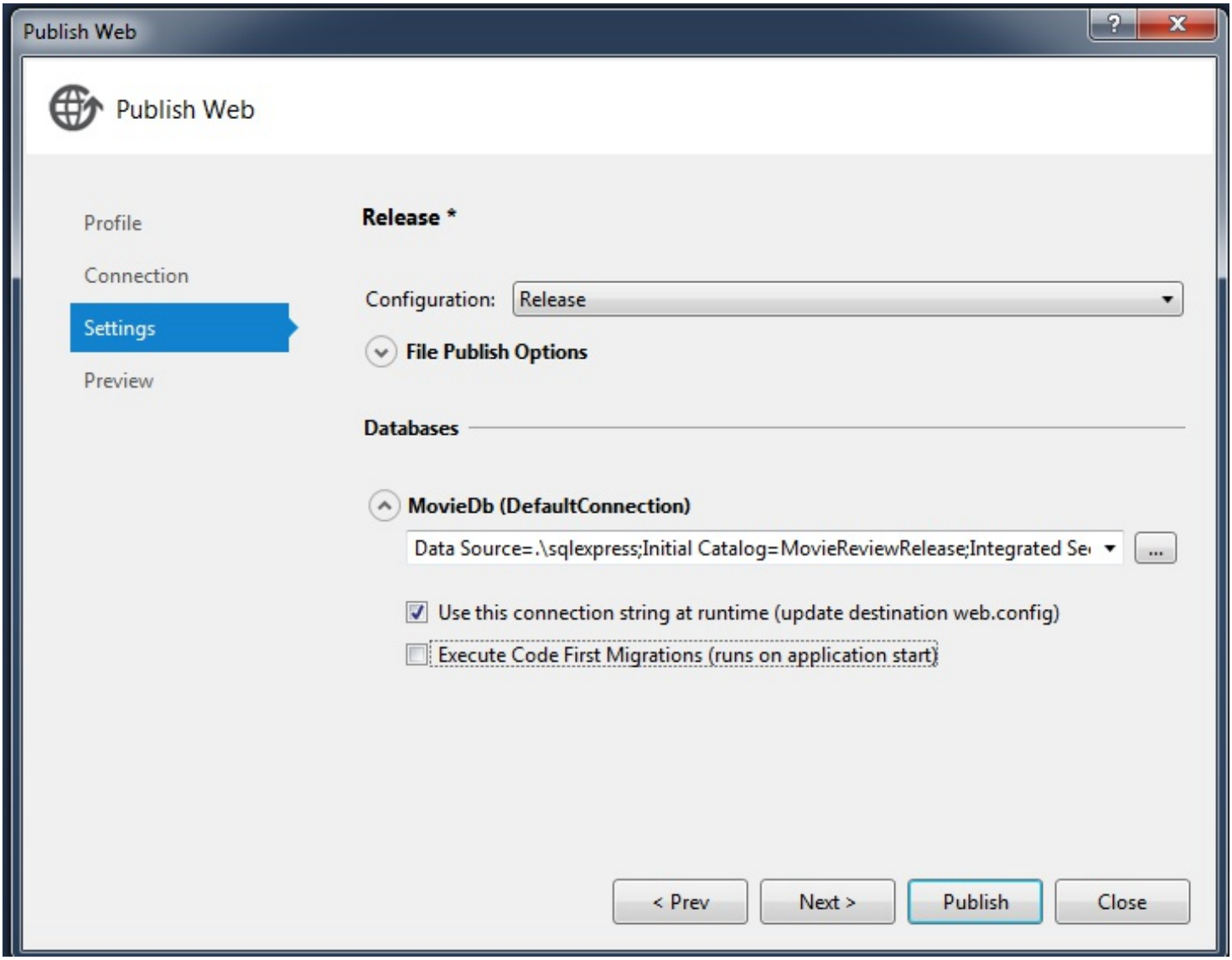


Then, I'll go ahead and click on next button, it will ask whether you want to deploy in release or debug mode, I'll select release mode. And then I'll also assign a connection string as default connection. So, currently the default

connection is the one which is there in "***web.config***" file. So, I can specify the same by using the helper button just after the text box in connection string setting with the below mentioned parameters.

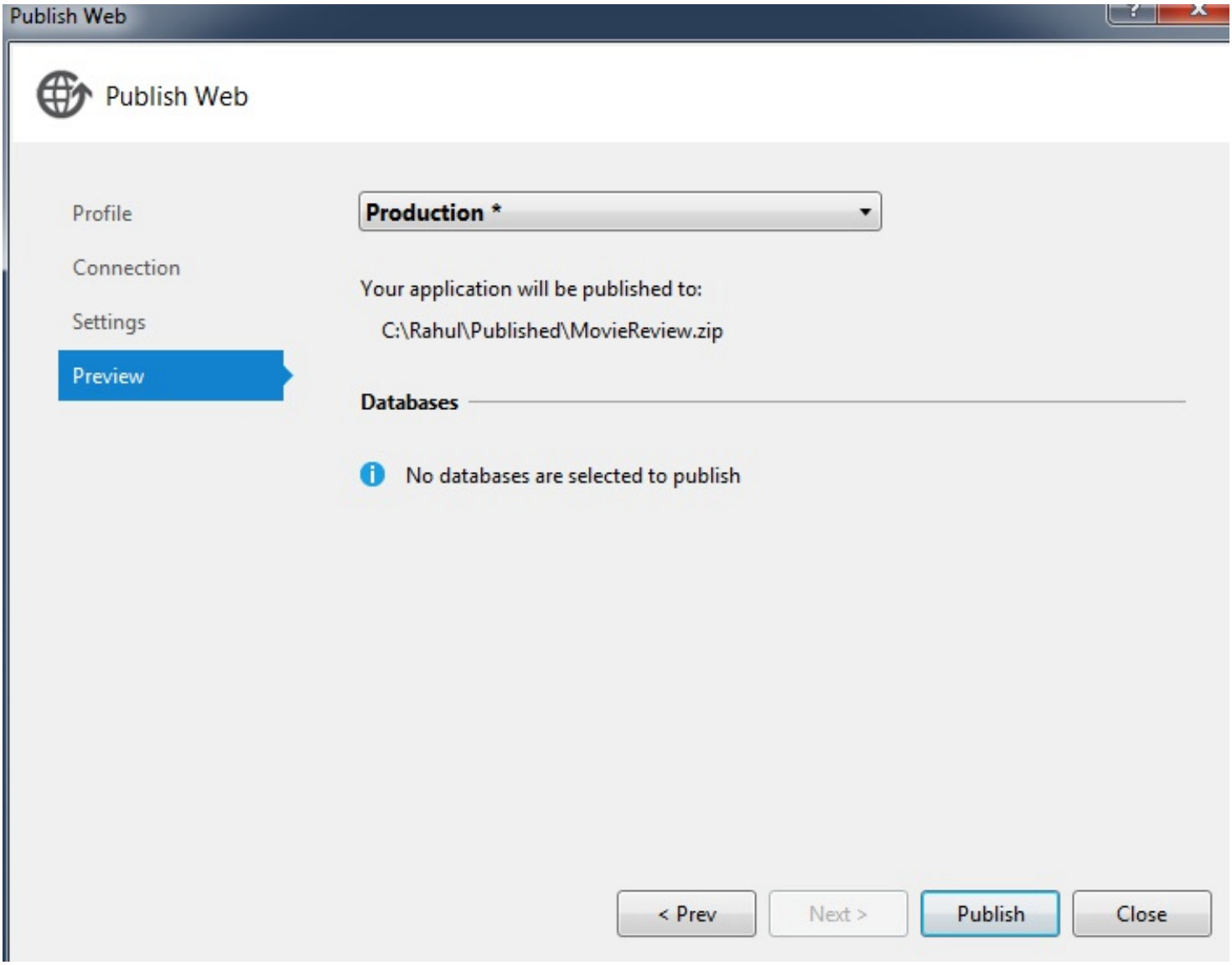

Destination Connection String
Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.
Data source:
Microsoft SQL Server (SqlClient)
Change...
Server name:
.\sqlexpress
Refresh
Log on to the server
Use Windows Authentication
Use SQL Server Authentication
User name:
Password:
Save my password
Connect to a database
Select or enter a database name:
MovieReviewRelease
Attach a database file:
Browse...
Logical name:

Now, I say ok. Then, it will take the effect.

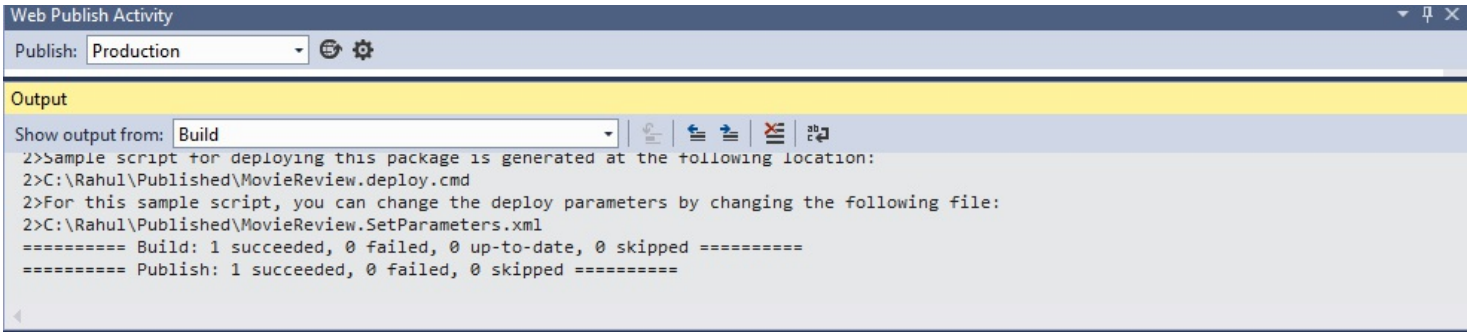


Now, when the application is deployed, the connection string will be substituted and replace the same what we have been

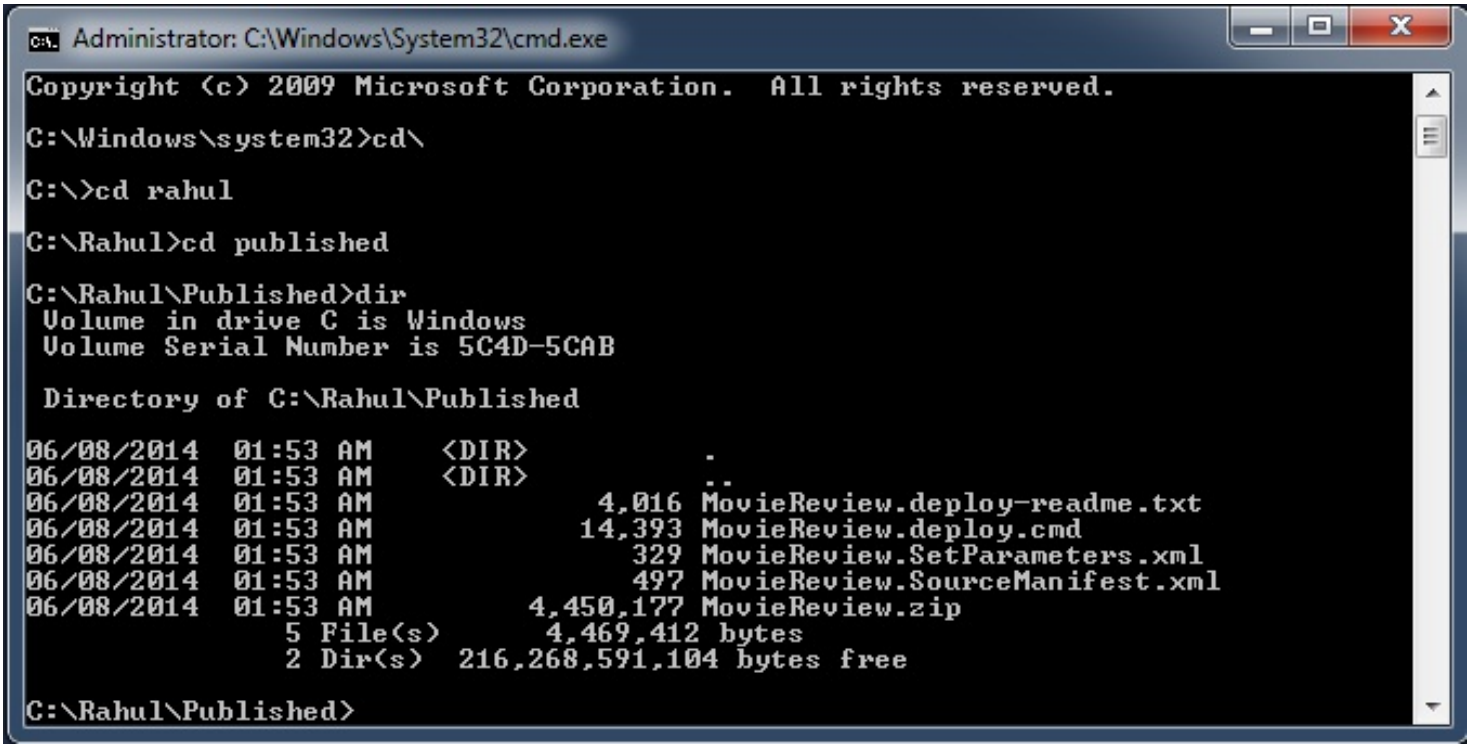
using for our development. Then, you can also select the next check box, which will execute the code first migrations from the config file. But, since we have taken care this explicitly from our code base, hence I keep this checkbox unselected. Now, when I click on next screen, it will give me preview screen as shown below.



Then, I'll click on Publish button. Then all the files get packaged up together, and upon successful publish, it will produce the result in the Visual Studio itself.

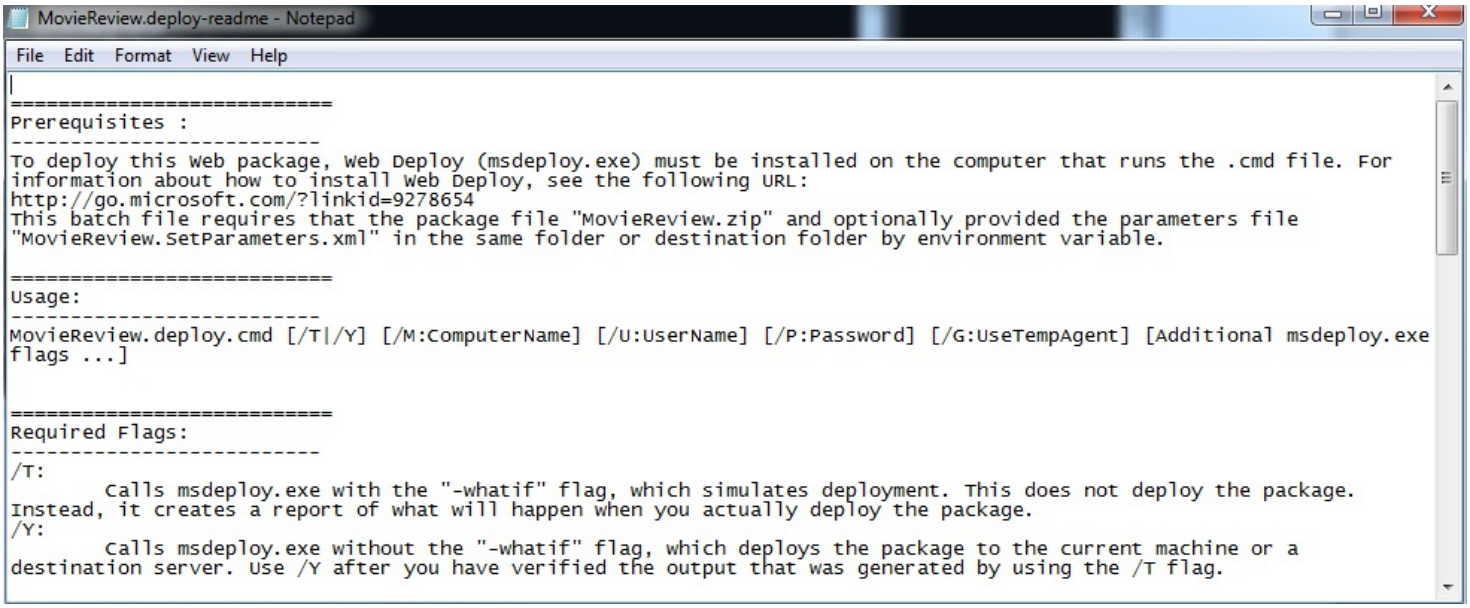


Now, I'll open the cmd prompt in administrator mode and then navigate in the directory wherein I placed the zip file as shown below.

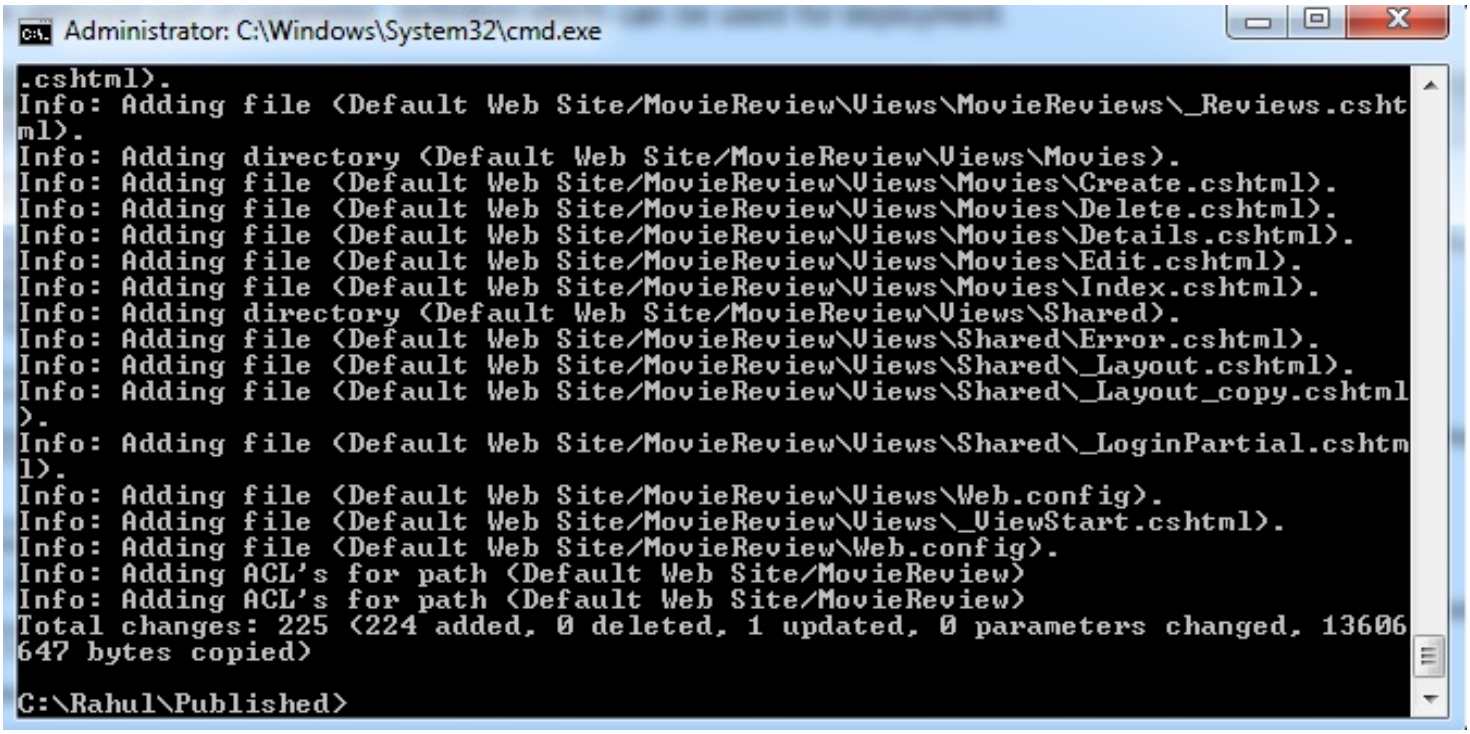


So, when I type the command

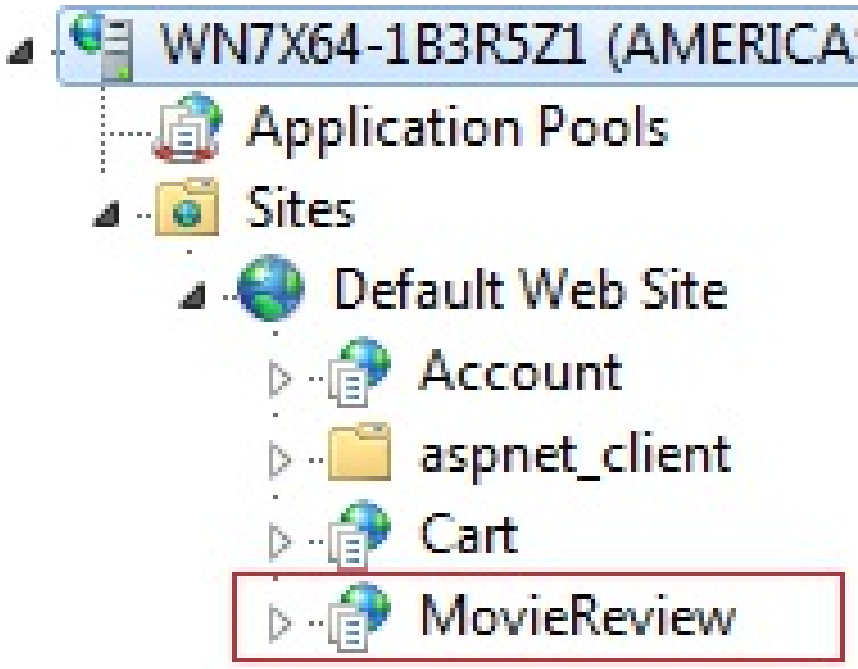
"***MovieReview.deploy.cmd***", then it will open a little help window wherein it will have all different sort of parameter definition which can be used for deployment.



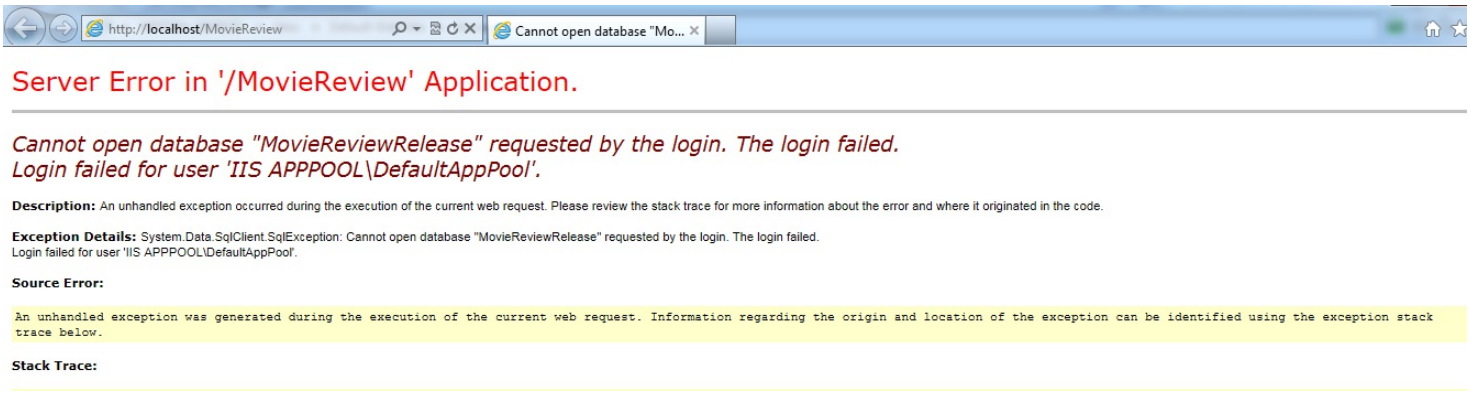
Now, if run the same command with the parameter "***/T***". It will just check whether everything is perfect or not, whether any error is there or not like shown below.



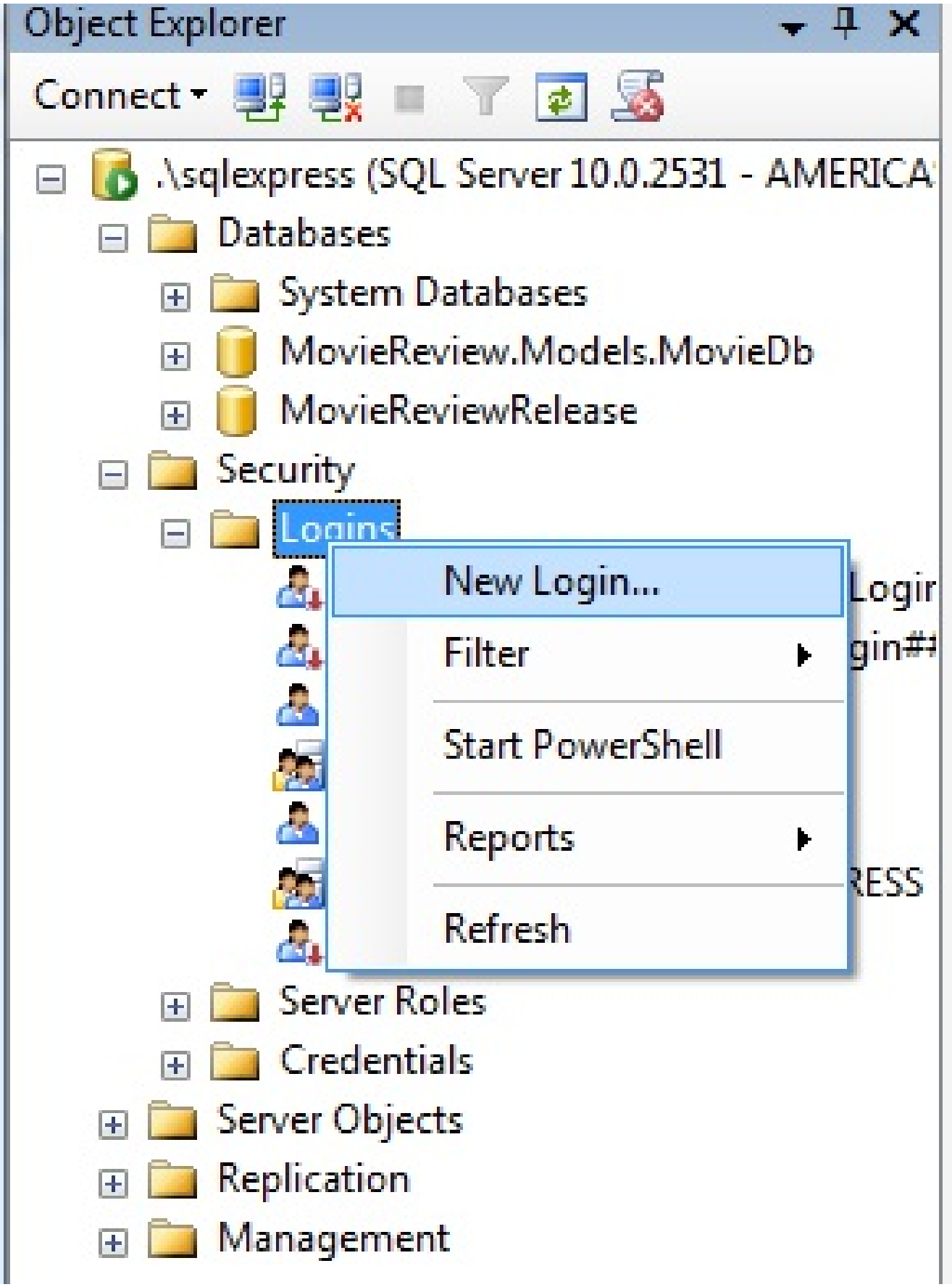
Now, everything looks fine, so I can go ahead and run the command using "***/Y***" so that it will deploy the same on the web server. Now, once this is done successfully, then it will add one new site at the specified location as shown below.

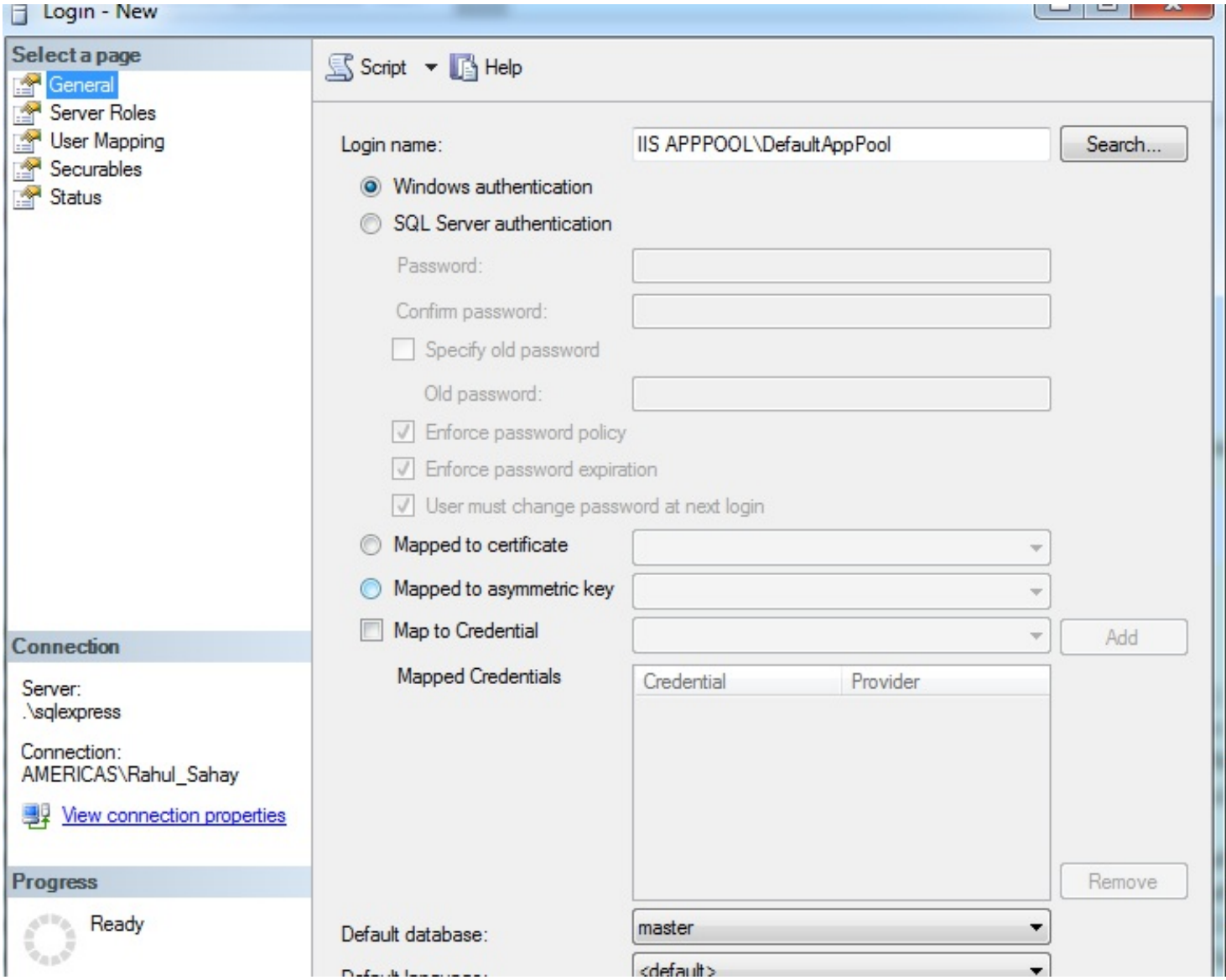


Now, when I navigate to this site, then this will produce me the below error.

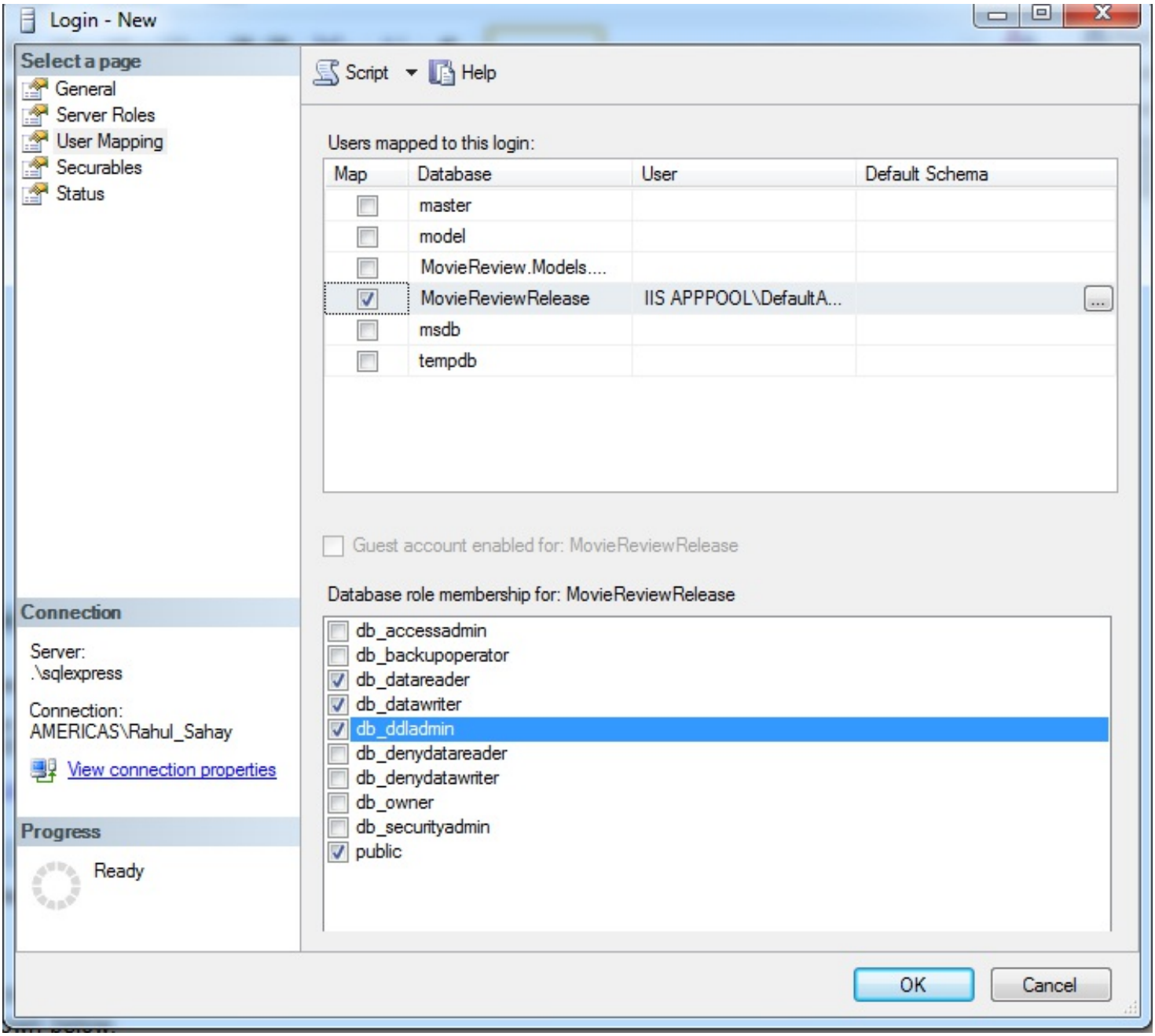


Now, my application is trying to connect to the database, but couldn't get there, as it doesn't have the permission of the same. And this is because the IIS worker process runs under the identity "***IIS APPPOOL***". And, currently this user is not having the permission. So, to grant the same, I'll go to the "***Sql Server Management Studio***", I'll create a new Login for this account and grant permissions to this as shown below.

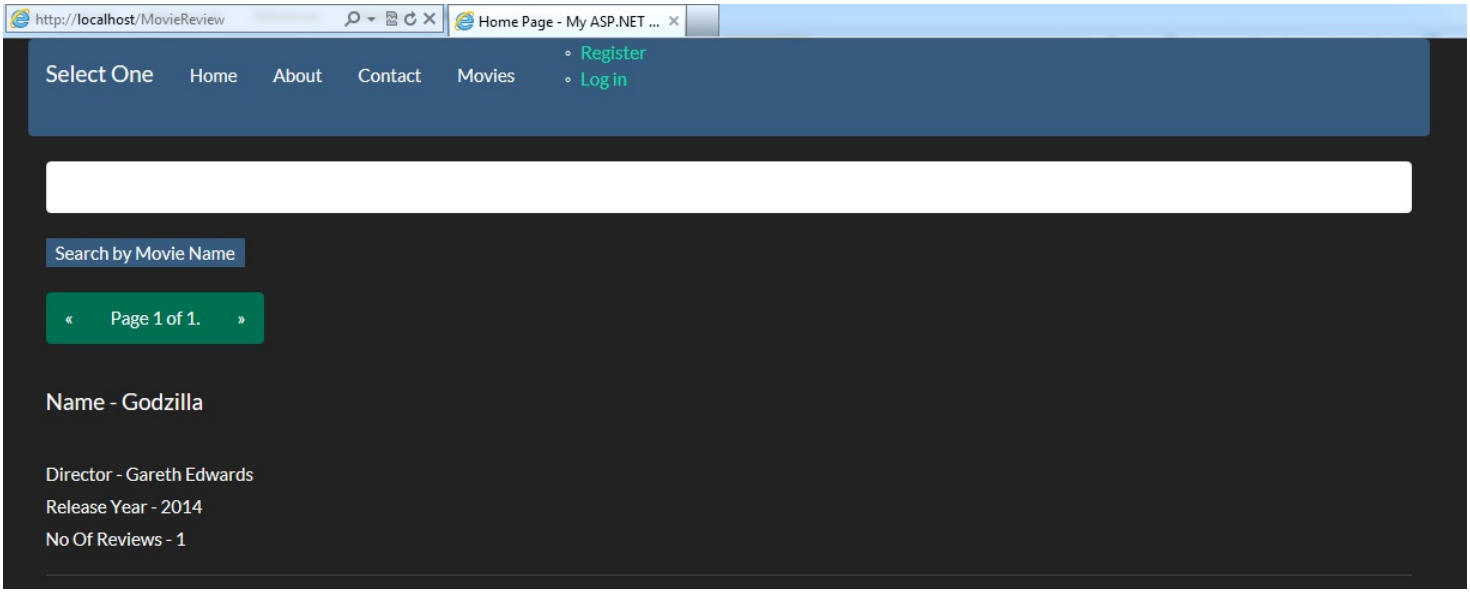

Object Explorer
Connect
.\sqlexpress (SQL Server 10.0.2531 - AMERICA
Databases
System Databases
MovieReview.Models.MovieDb
MovieReviewRelease
Security
Logins
New Login...
Filter
Start PowerShell
Reports
Refresh
Server Roles
Credentials
Server Objects
Replication
Management



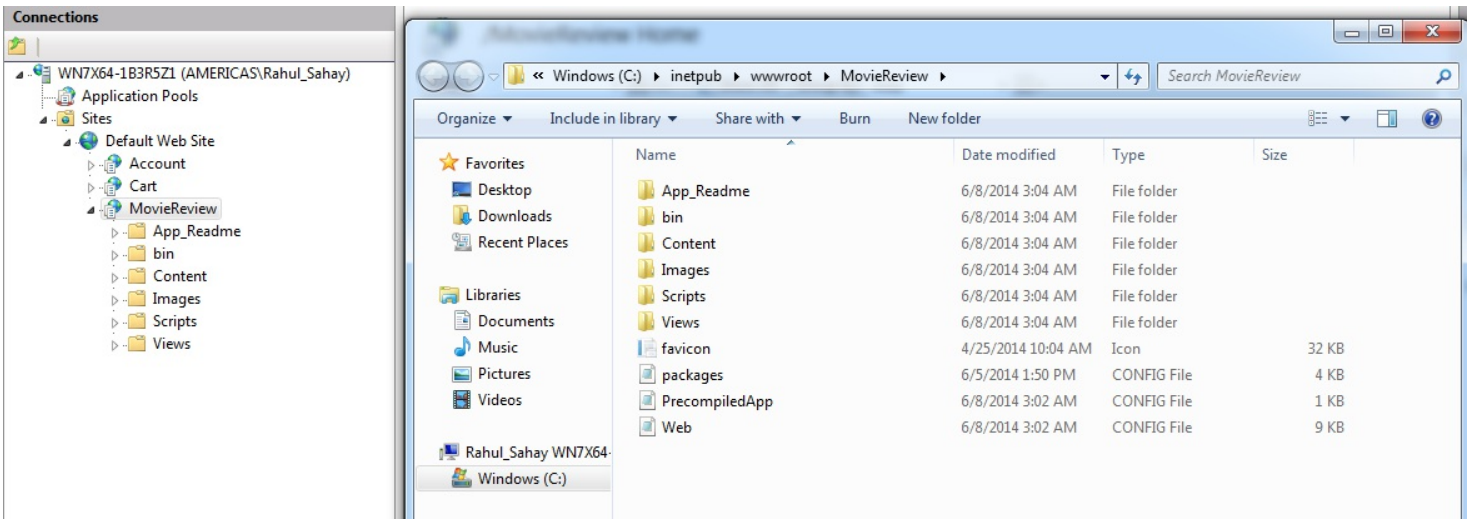
Then, I'll come in the User mapping tab and select the newly generated db and then grant the following permissions to it as shown below.

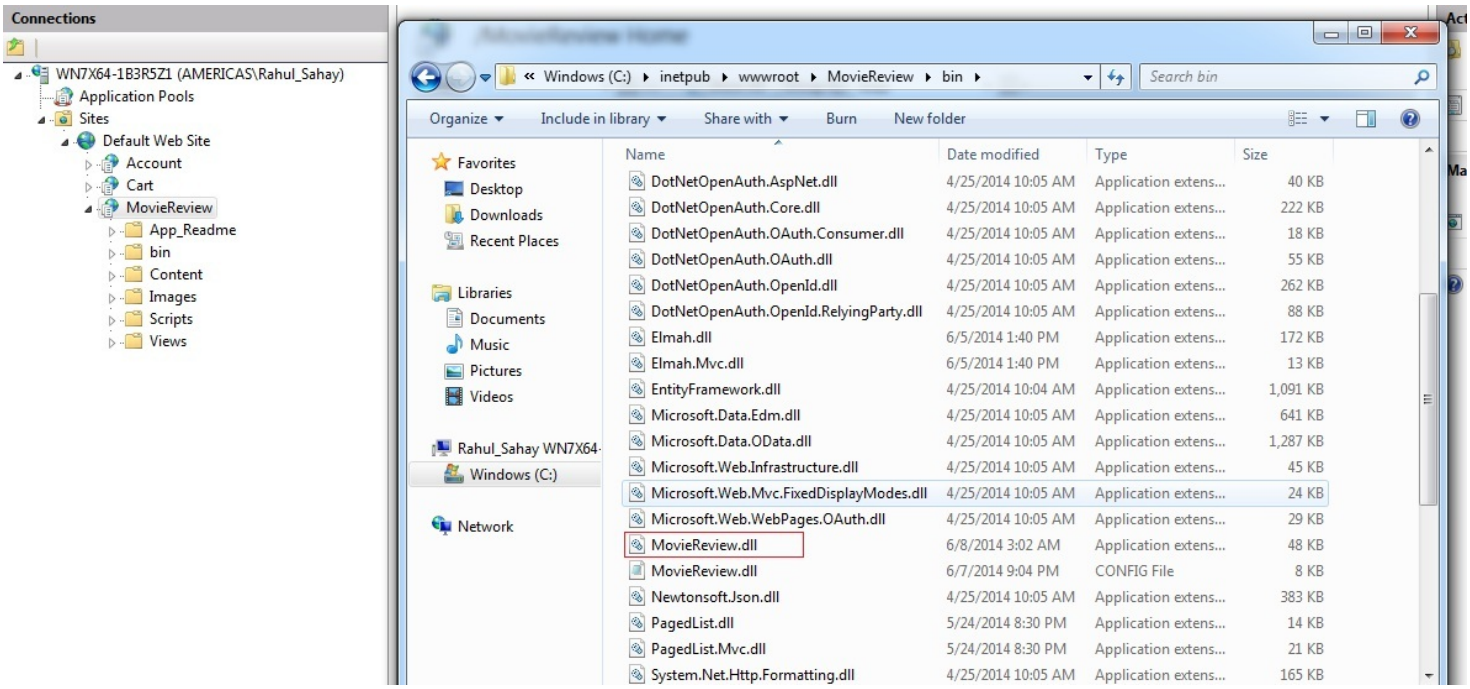


Then click ok. Now, when I come back and refresh the browser, then it will produce me the below result, now with valid movie names but this time from site hosted on IIS.



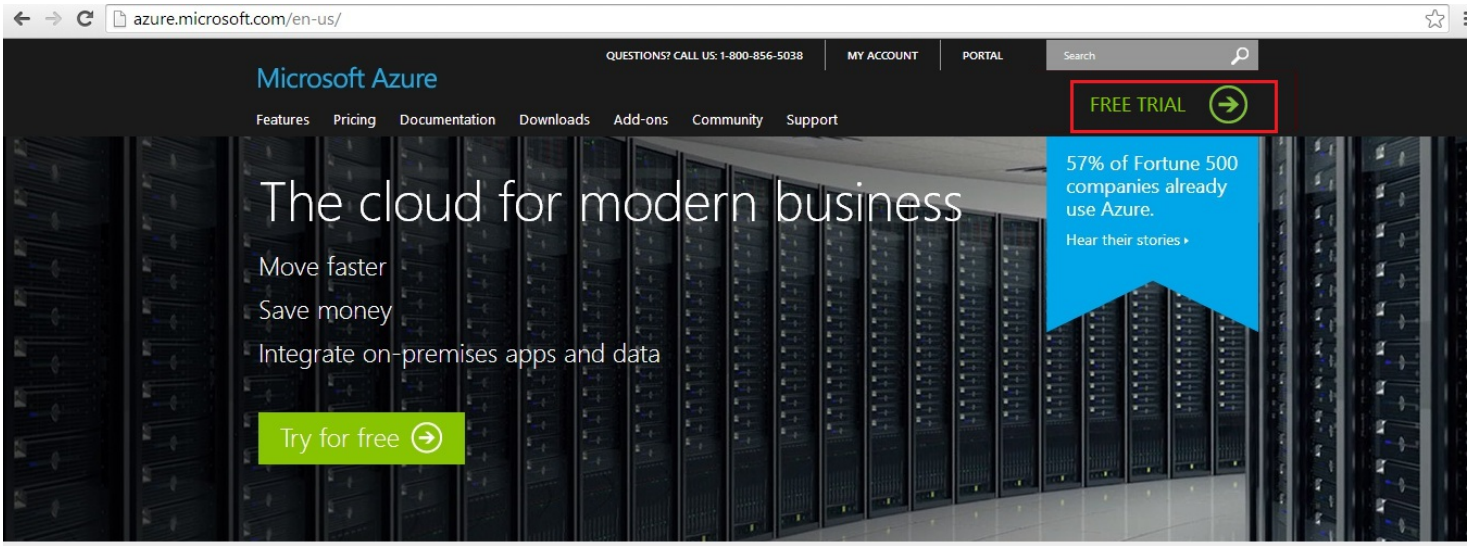
Now, if you see the folder in IIS, you will see that it's not having the C# code or Controller's code, reason being that all application logic is compiled and embedded in the dll which sits in the bin folder as shown below.



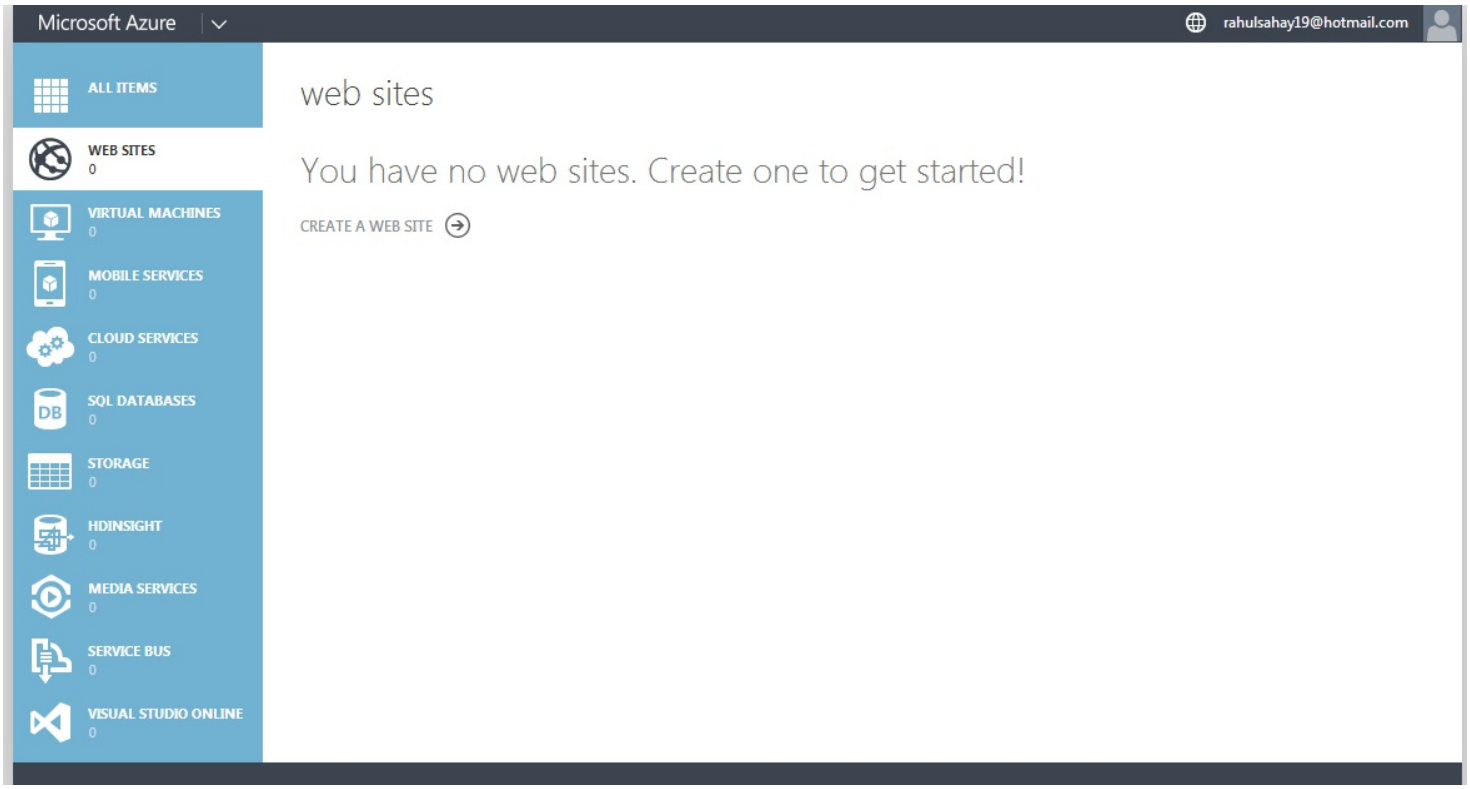


Deployment on Azure:-

Now, in this section, I am going to deploy on Microsoft's Cloud and that is Azure. Now, to get started I'll navigate to "http://www.windowsazure.com" and this will produce me the below page.

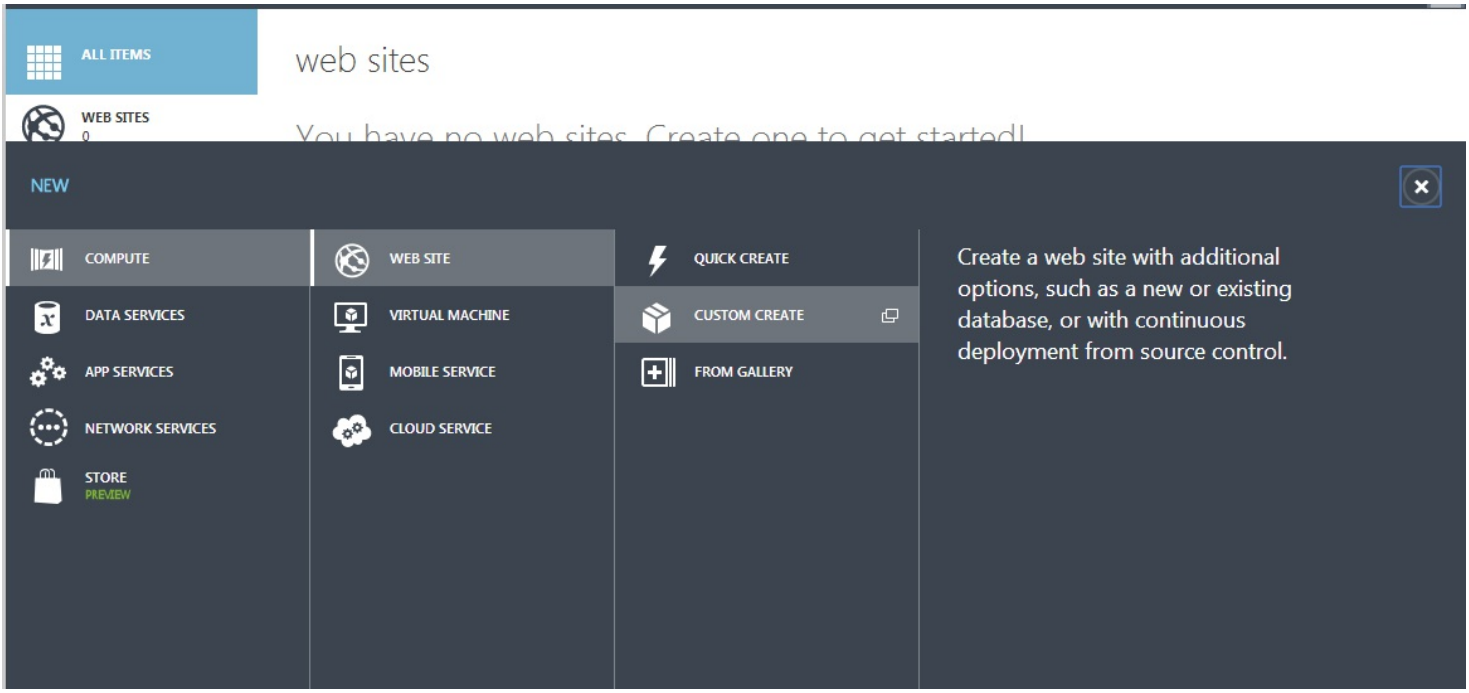


Now, there are two things, One is Free Trial which I highlighted above. So, this is the option for 1st time users, where in Microsoft is providing its Cloud services for free for a month. And for registered users, you can use the **Portal Link** shown above. Now, once logged in with your Microsoft Account, you will land up in the dashboard page.



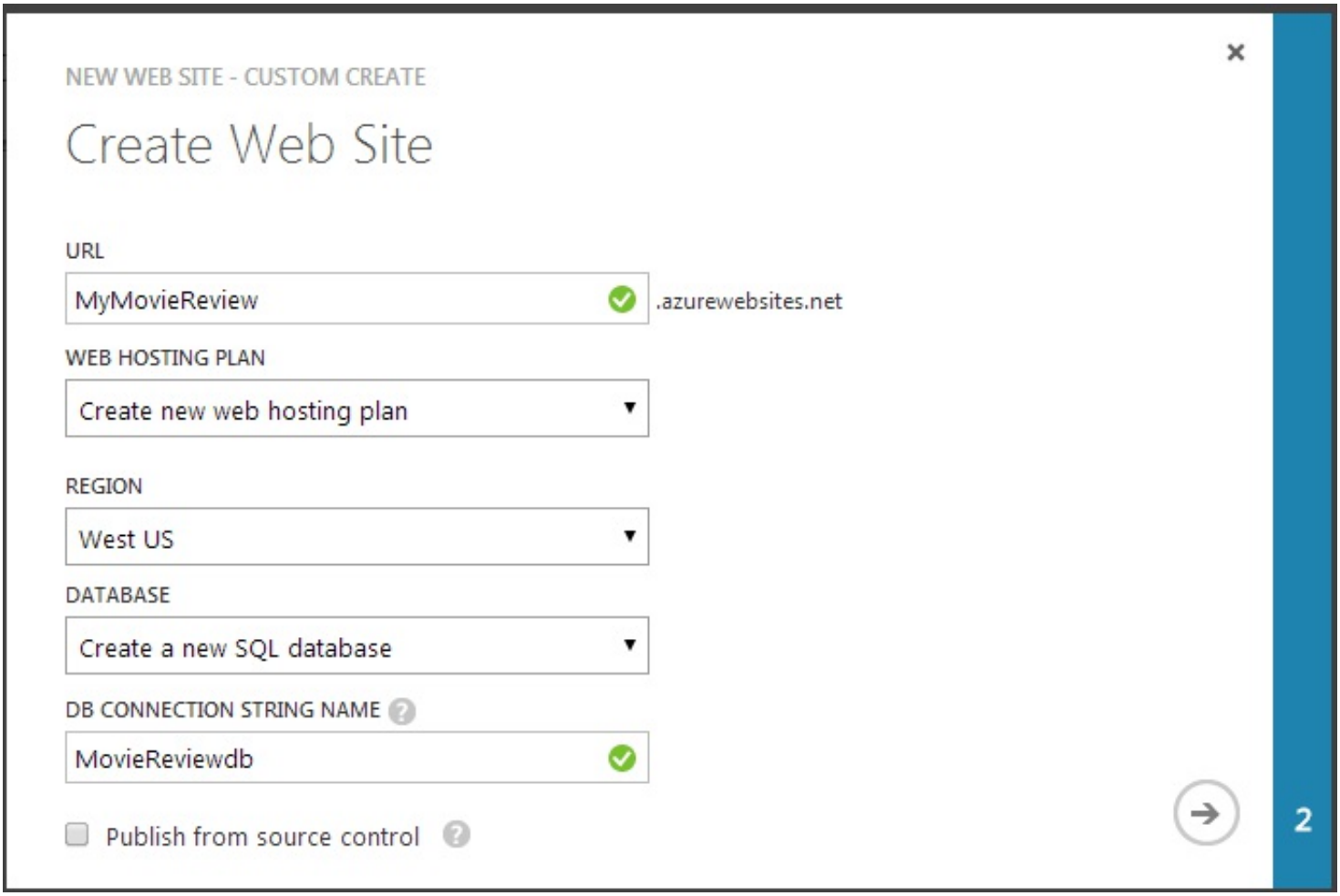
So, as you can see in the dashboard page, you could host multiple things in cloud platform. So, the intent of using cloud is start small and then scale big. So, basically you are going to billed for the actual resources you are using. And this tool is really nice. It gives you the complete picture of your site utilization. So, as you can see in the above screen shot in the side menu. You can have

Virtual Machines, **Host Mobile Services**, **Cloud Services** then **SQL Databases**, **Media Services** and many more things. So, here I'll go ahead and create a new website with a database as shown below.

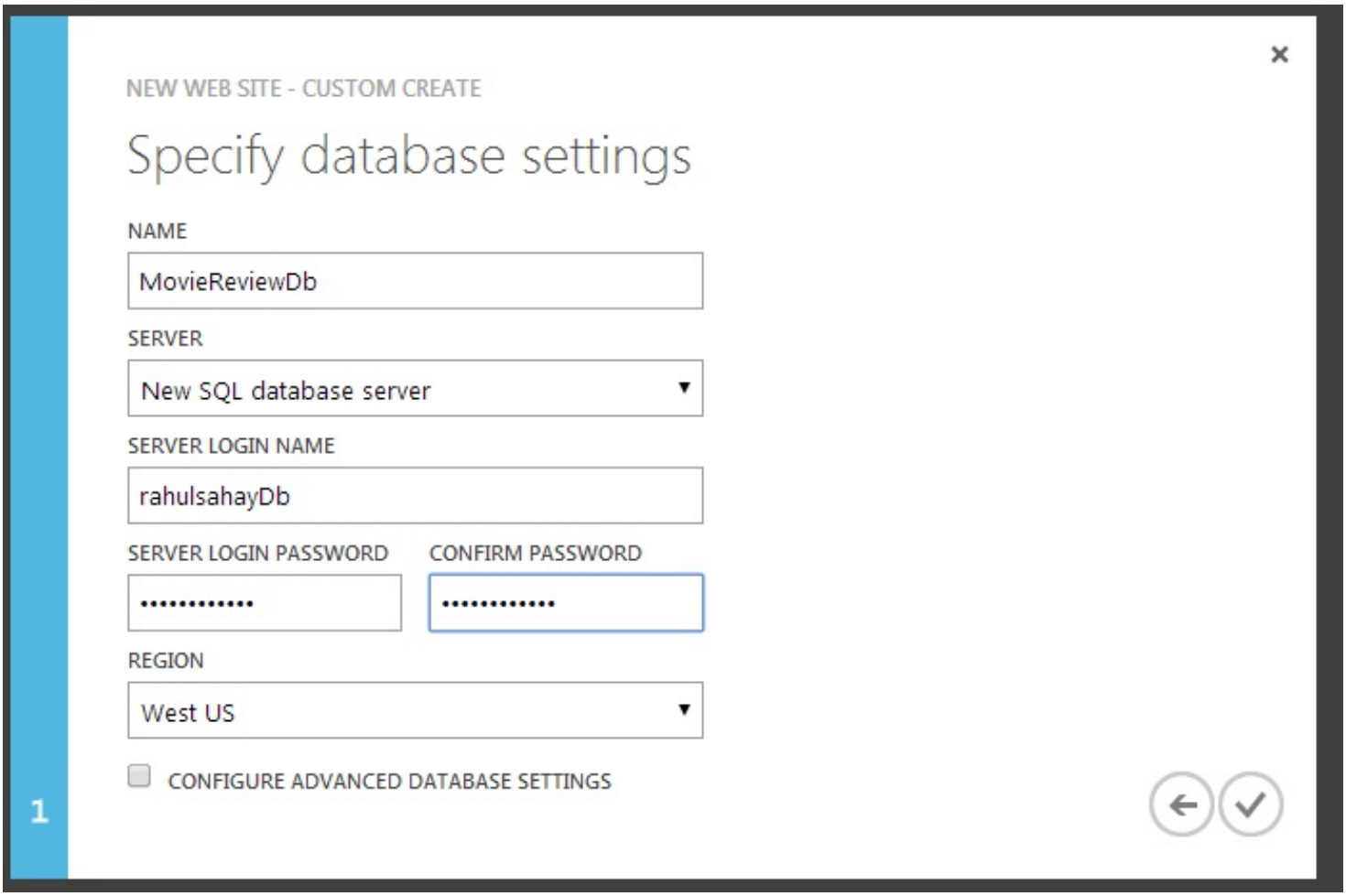


Now, I'll provide the following parameters in there. Currently, I am using "***AzureWebsites.net***", but I can change this by just providing the "***DNS***"

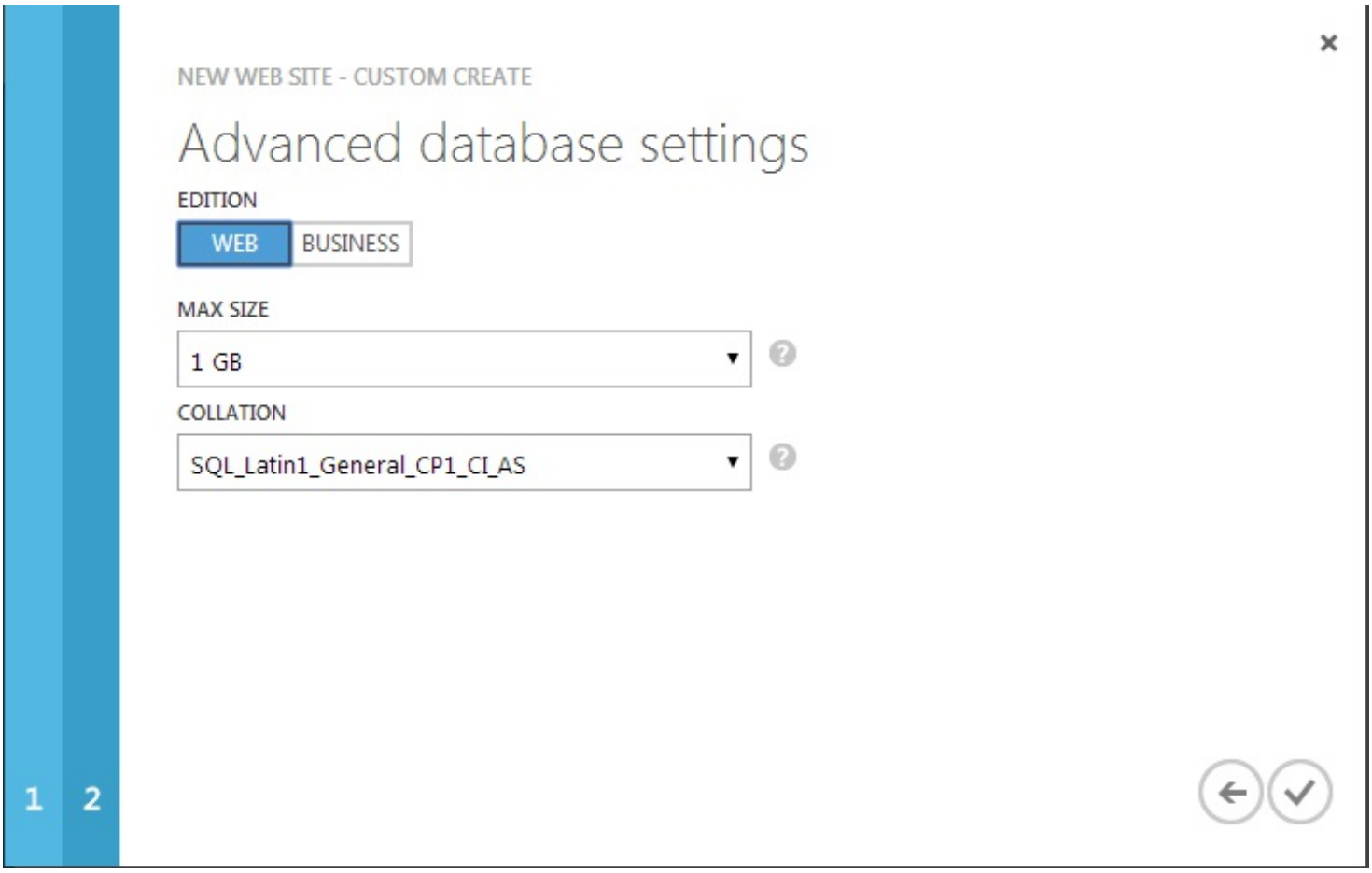
of my hosting provider. But, for demo purpose that's ok. Then I am going to select the region for hosting provider and in the end I'll create a new SQL Database for my site from the dropdown.



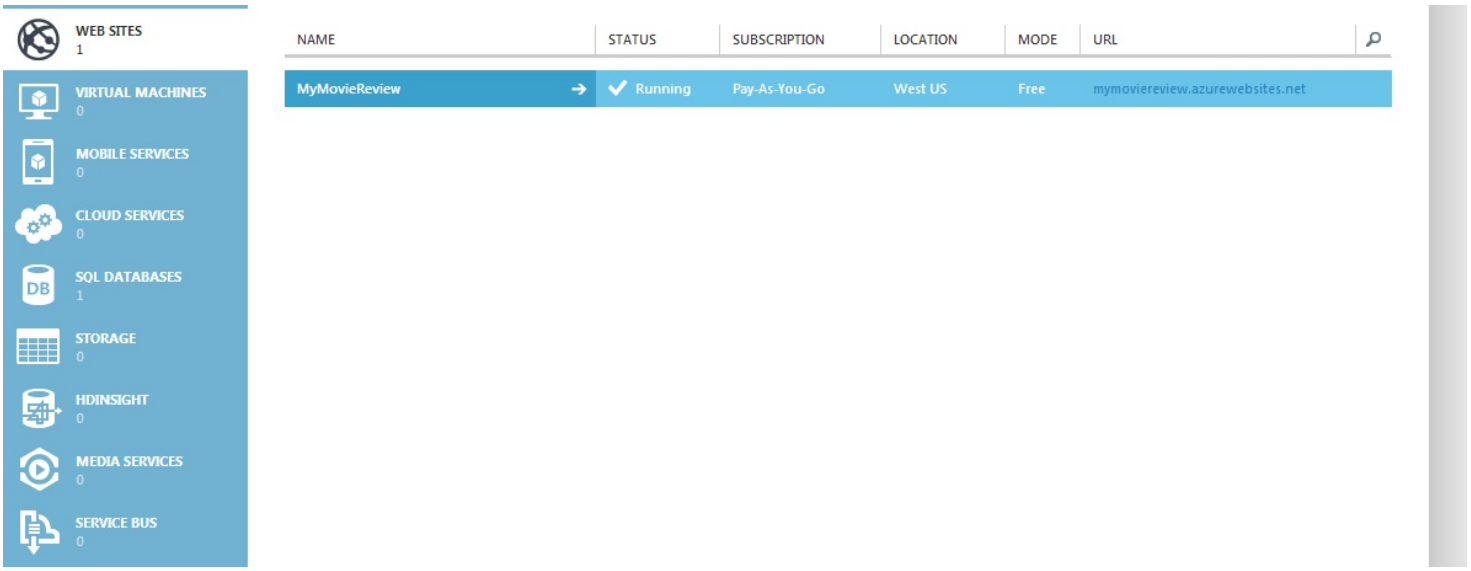
Then, I provide the database settings as shown below.



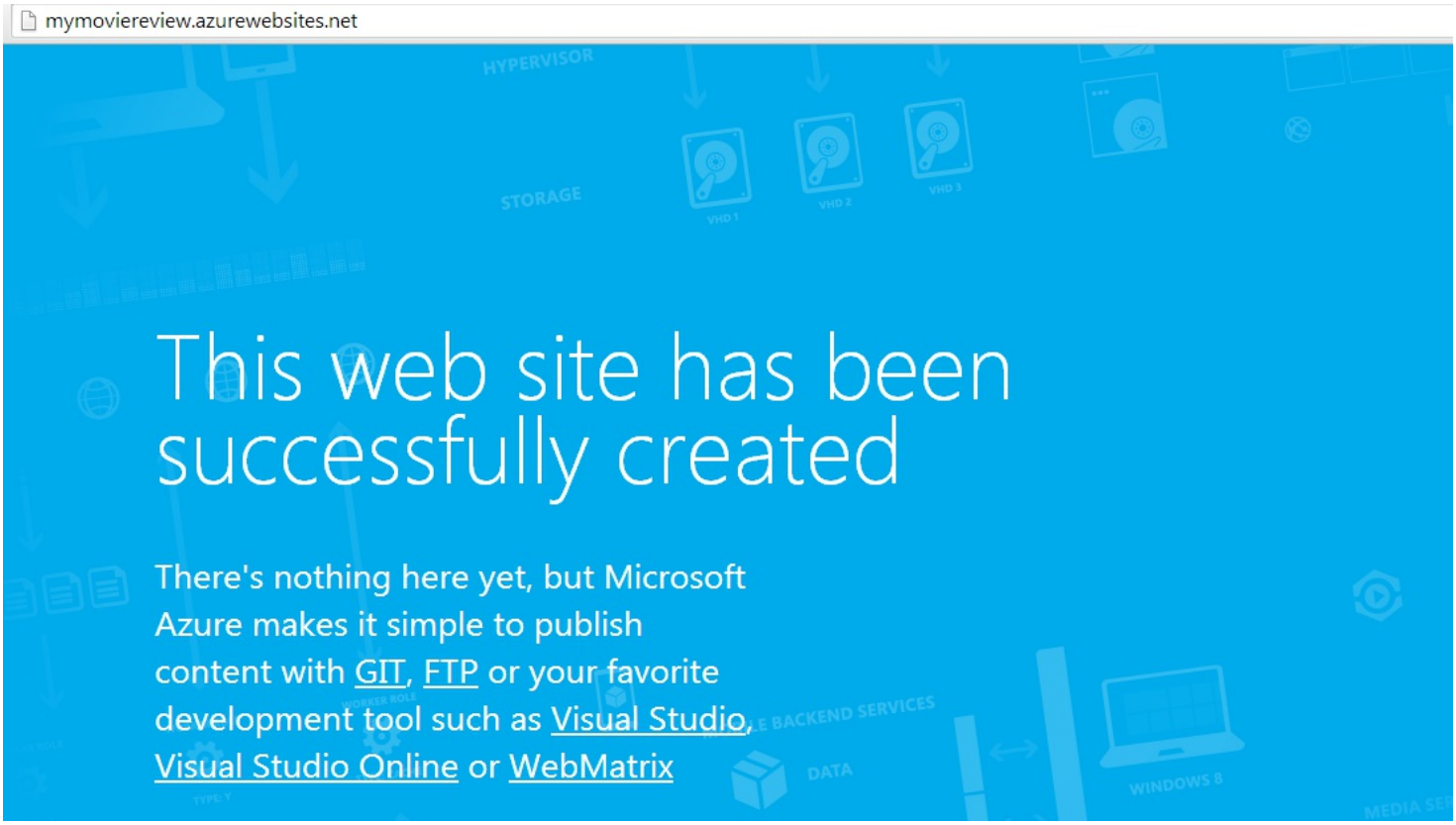
Now, as you can see in the screen shot I have created a new SQL Database Server with the above mentioned parameters. Now, I can also configure advanced database settings, by selecting the checkbox shown in the screenshot.



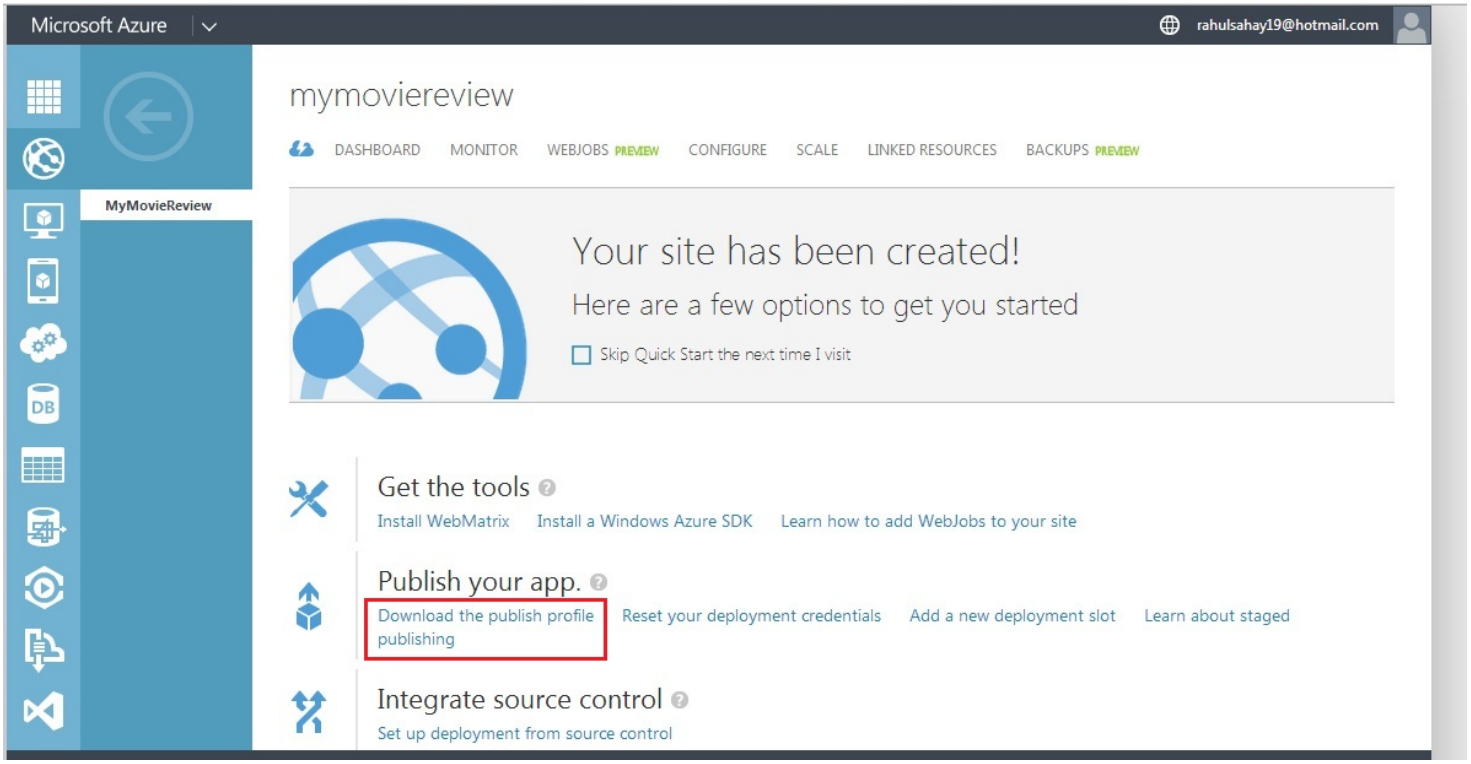
Now, this is the place where in I can go ahead and select my database size, what edition I am targeting means whether it's a **Web Edition** or **Business Edition**. Now, with all these settings in place I click on check mark to complete the settings, then it will produce me the below result.



So, you can see here, that my website is created with the default setting, now I can browse this as well by clicking on the URL link as shown above, but it will produce me the default page as shown below.

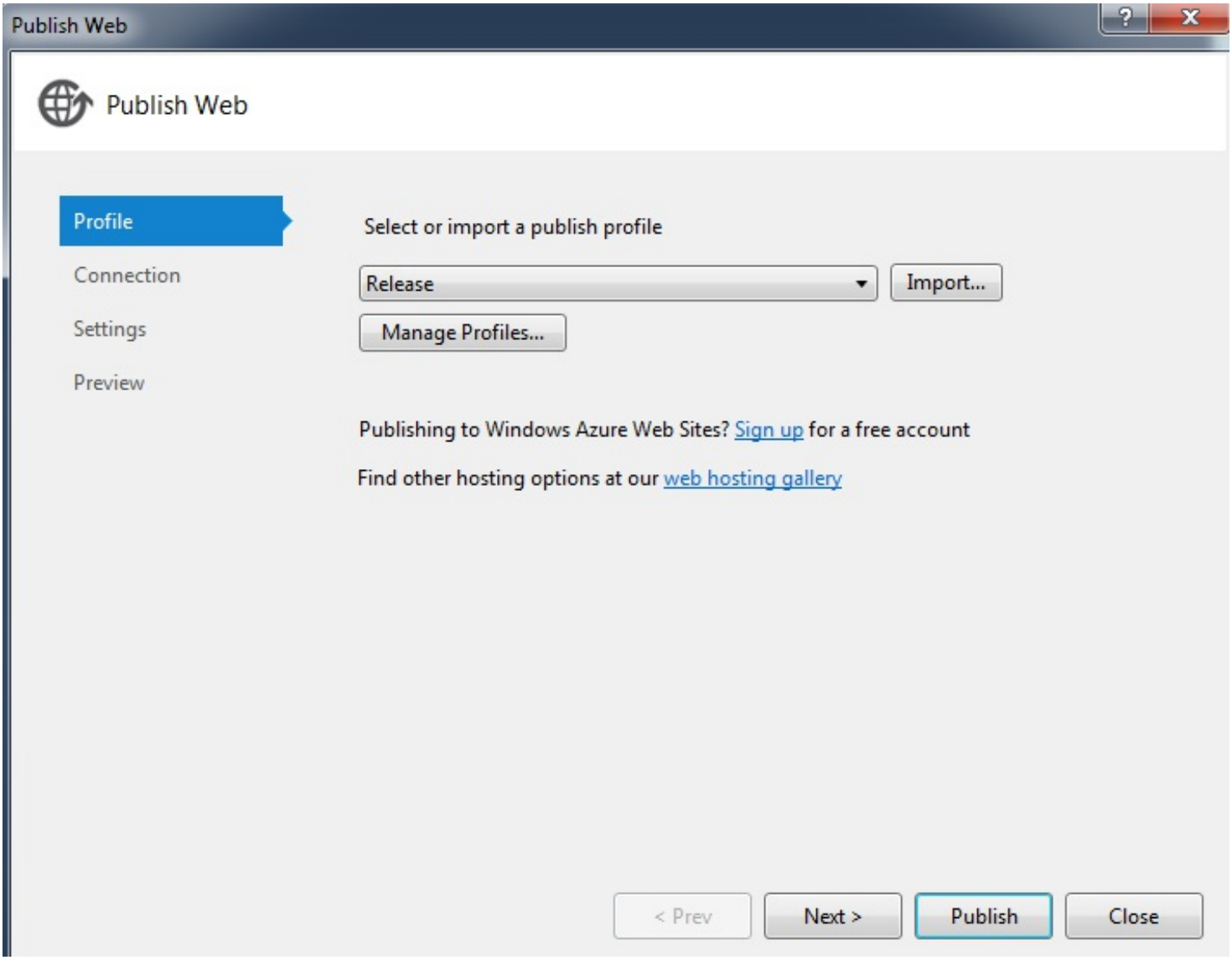


Also, if you click on the link as shown in the dashboard page, then it will open me a descriptive link for my newly created setting.

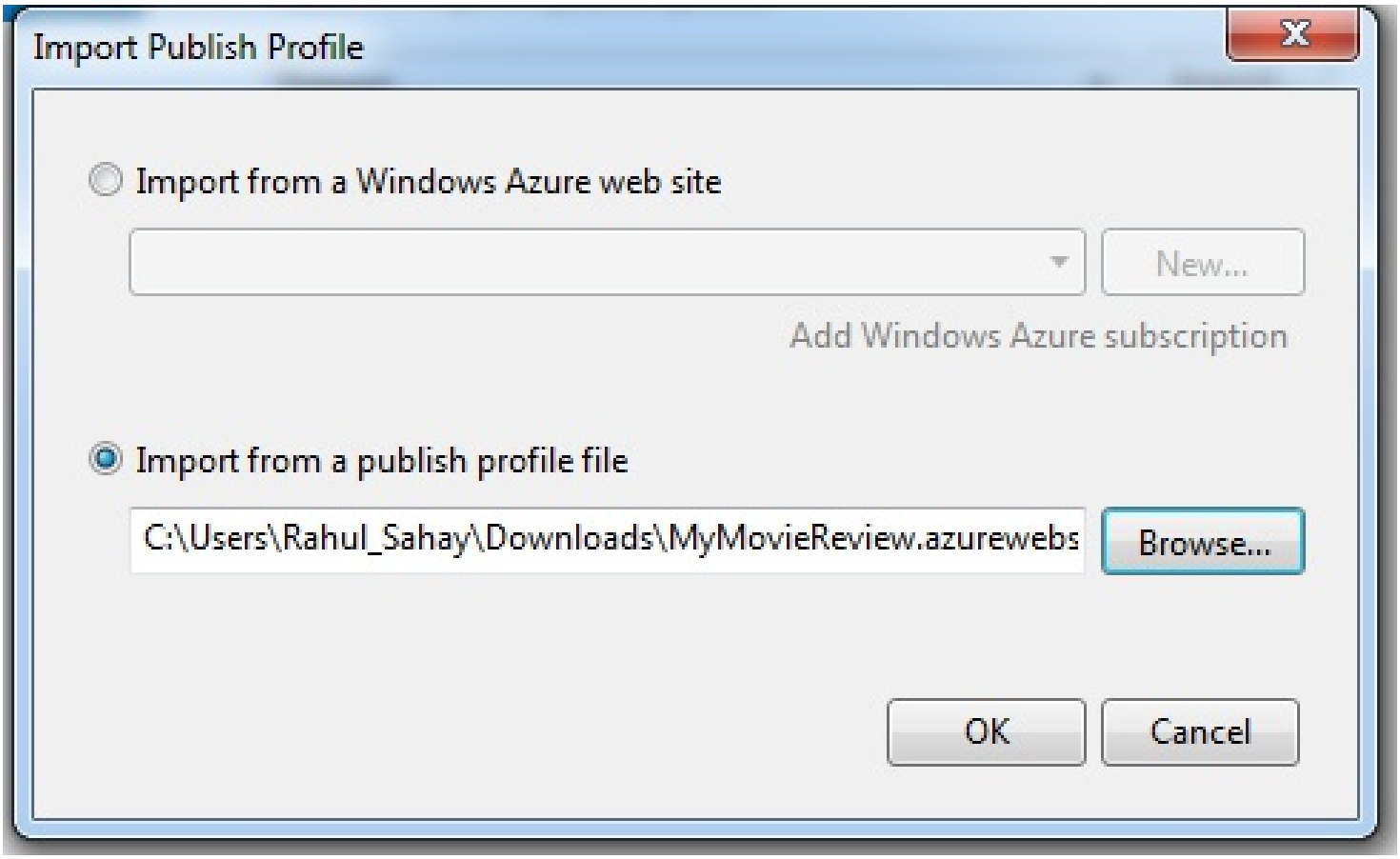


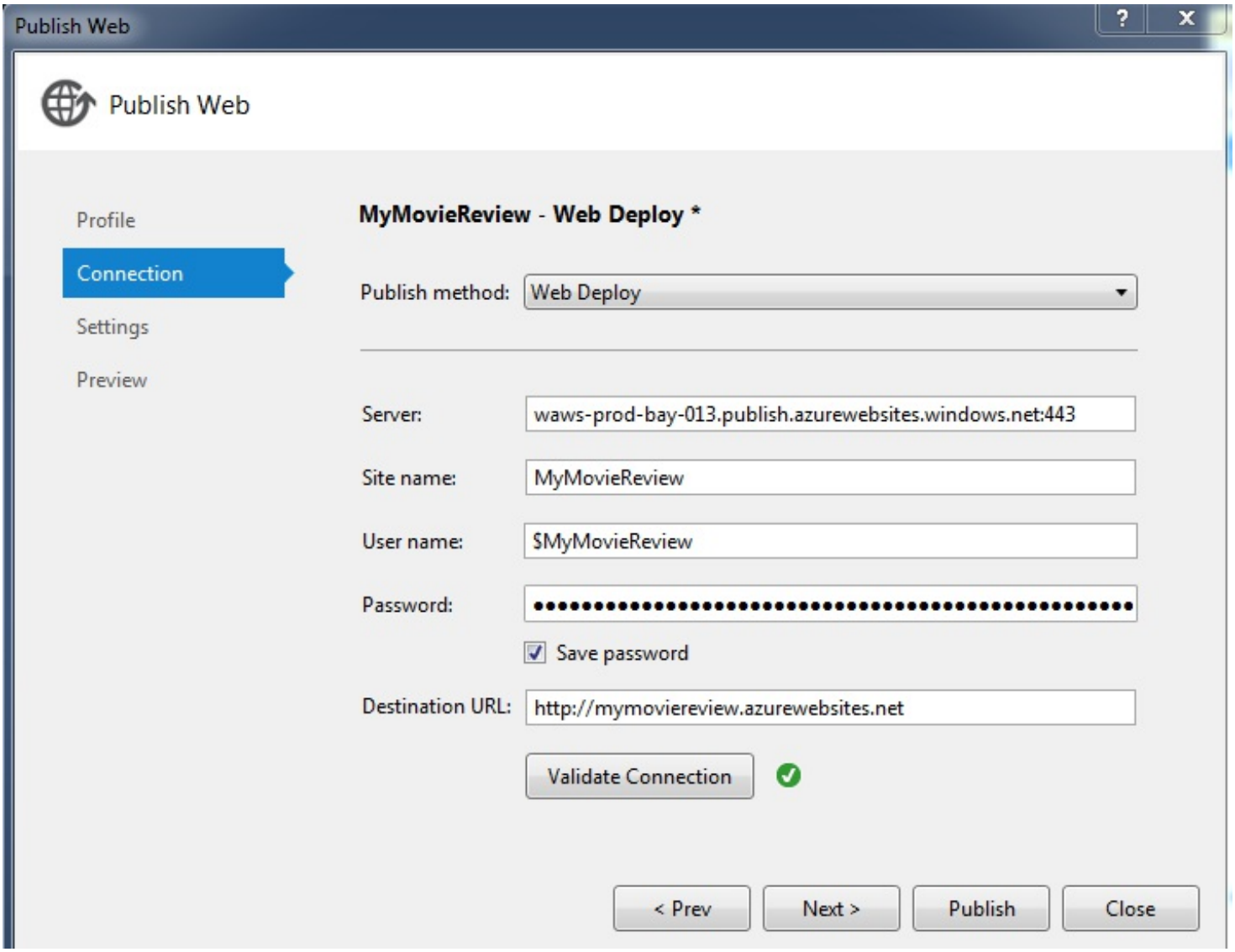
Now, as you can see here in the above screen shot there is a link, which is having all the settings related to your newly created site. So, when you click on this link, it will download one big xml file, which will have all the settings related to your website. Now, for publishing your site, you need to go into Visual Studio and click publish and this time you need to choose that xml file

which you have downloaded as shown below.

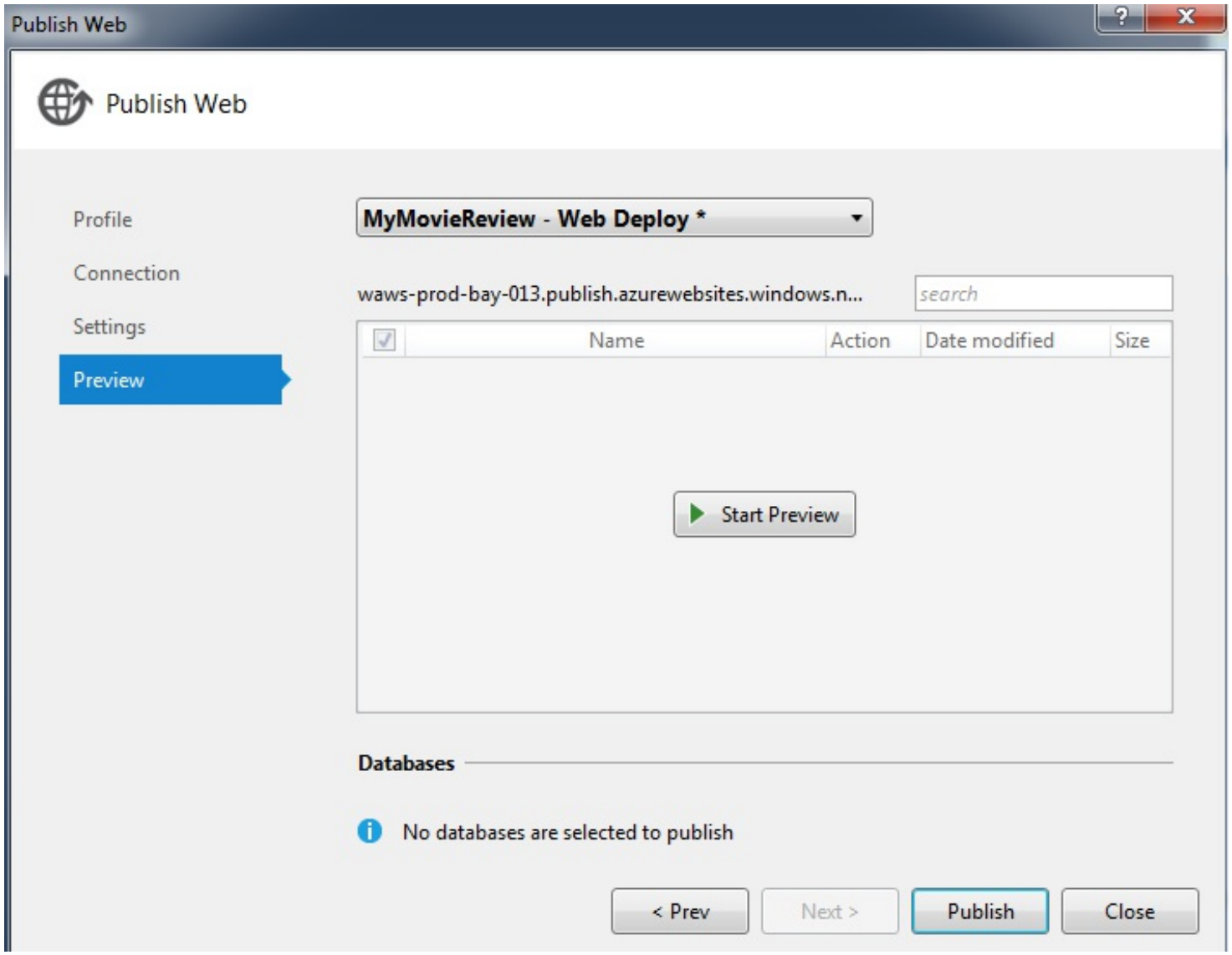


Now, click on the import button next to the drop down and then it will have all the required settings for Azure in there.


Import Publish Profile
Import from a Windows Azure web site
New...
Add Windows Azure subscription
Import from a publish profile file
C:\Users\Rahul_Sahay\Downloads\MyMovieReview.azurewebs
Browse...
OK
Cancel

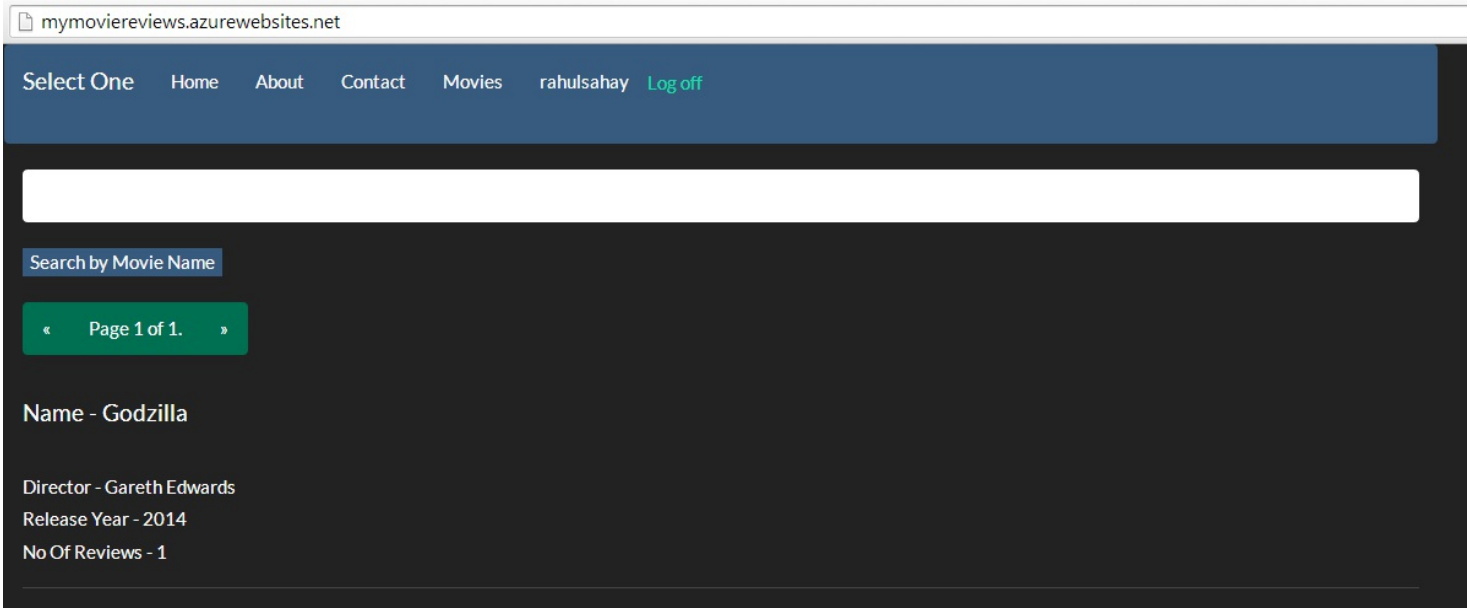


Now, when I do next it will provide me the settings page. Now, in this page I don't want to change any setting, so I'll just go and click next. Then I'll click next to go in the preview page as shown below.



So, here I can do preview as well. But, that's ok. I'm hoping everything will be fine, hence directly clicking on publish button. Now, since this is going on internet, hence this will take some time. Upon successful completion when I go

ahead and refresh the site, it will produce me the desired result.



So, as you can that now your app is live and ready to be exposed to internet users.

Note:- I have changed the URL later after Final Deployment as you can see in the above screen shot as this looks more meaningful. So now the URL is (http://mymoviereviews.azurewebsites.n

Also, later on, I may take this site offline as I hosted this only for demo purpose. Also, you can download the same code from **GitHub** (https://github.com/rahulsahay19/Movie-Review).

Summary:-

So, in this section, we have seen various configuration related settings, how to override the same. How, the hierarchies of these settings follow the path. We have also seen how to use custom settings in our application. Then, we have made some deployment related changes to our application and then

deployed the same on IIS 1st and then on Azure. So, ultimately we have covered two deployment options one for intranet and the other one for Internet.

Chapter 11: Routing Basics

WHAT DO you find in this CHAPTER?

⇒ Introduction

⇒ Route Definition

⇒ Areas

⇒ Ignore Routes

⇒ Debugging Routes

⇒ Constructing URLs from Route

⇒ Route Data

⇒ Route Constraints

⇒ Summary

Introduction:-

As promised earlier in the starting chapters, that we'll delve under the hood

of Routing, So, we have come to routing chapter to understand the routing principle and why this is one of the most important topics which need special attention. So, one important point to note here is that routing is not only limited to ASP.Net MVC as we discussed already while creating our 1st controller.

Now, as we progresses we have understood the fact that file system is not the case with MVC based frameworks. So, basically in these cases, it takes certainly a different approach by mapping URLs to controllers' methods rather than some physical files. Routing in ASP.Net MVC Framework basically serves two main purposes.

- **It matches the incoming request which is going to be matched against one specific controller's action.**
- **It also constructs URLs which corresponds to controller actions.**

Now, one more important concept I would like to clear here that Routing is different from **URL Rewriting**. So, **URL Rewriting** is basically focused on mapping one URL to another URL. This approach is basically applied at those places where in user is converting old URLs to new ones just to make the URL user friendly. On the other hand routing is entirely focused on mapping a URL to

a resource. Another difference in this case is that Routing also helps in generating URLs using the mapping rules that it uses to match the incoming URLs. On the other hand, URL Rewriting only applies to incoming request and doesn't generate any new URL.

Route Definition:-

Any ASP.NET MVC application needs at least one route to define how the application should work. Now, in large enterprise application, one could have multiple routes defined in their application. In this section, we'll discuss how to define routes. Route definitions

start with the URL pattern, which specifies the pattern that the route has to follow. Along with the route URL, routes also specifies default values and constraints for the various parts of the URL.A route can also have a name associated with it.

So, as soon as we create any ASP.Net MVC application, you will notice in the "***Application_Start***" event, there is a call to register the routes. So, when you inspect this, this will open a file "***RouteConfig.cs***" under "***App_Start***" folder. So, inside "***Global.Asax***" file "***RouteConfig.RegisterRoutes(RouteTa***" there is a call using "***RegisterRoutes***" method which is registering all the

routes for the application. So, now this method is moved in a separate file under "***App_Start***" folder as shown below.

```
public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");


            //Movie/Avatar
            routes.MapRoute("Movie","Movie/{name}",

new{controller="Movie",action="Search",name=UrlParameter.Optional });

            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Home", action = "Index", id =
UrlParameter.Optional }
            );
        }
```

Now, let's discuss "***MapRoute***" method here 1st. so, basically it takes the route name with URL pattern for the route. Now, later in this chapter we will discuss Route name separately. But for

now just focus on the pattern as shown below in the table.

URL	Parameter Values
http://localhost/MovieReview/Movies/Details/1	1st Parameter = "Movies" 2nd Parameter = "Details" 3rd Parameter = "1"

So, as you can see above the route code will parse the URL into a dictionary of keys and values which is stored in "***RouteValueDictionary***". Now, if you look closely the URL, it is composed of various segments. A segment is nothing but values between slashes (/). So, in the pattern matching rule, you can see that 1st parameter is picked up as **controller**, then **action method** and then **id**. So, here for this URL invocation my below method is getting invoked

```
//
        // GET: /Movies/Details/5

        public ActionResult Details(int id = 0)
        {
            Movie movie = db.Movies.Find(id);
            if (movie == null)
            {
                return HttpNotFound();
            }
            return View(movie);
        }
```

Again, this method is getting picked up by my "***Default***" route. Now, it is also possible that in my route definition, I don't have any definition for any action method, directly parameter, which is there in "***Movie***" route. So, it's acting there as a search filter based on the incoming parameter. This scenario we

have already discussed while designing the "***Movie***" route.

Now, there is also a definition for Defaults in the **MapRoute** section, so it basically tells, what would be my default values for the route, if user doesn't provide these values from their side. It includes **controller name**, **action method** and the **default parameter values**. Now, associated to these default values, we also have one important thing which is meant for default parameters. We are here supplying "***UrlParameter*.Optional". Although, we could have supplied parameter = "". So, in the 1st case, it won't add any value to the dictionary, but in later case that

parameter is added to the dictionary against the **empty string value**. So, the difference is pretty significant. So, at various scenarios you may land up in exception while extracting null value. However, the exact difference with demo already explained while designing route in previous chapters.

Areas:-

Now, this is again one of the important topics of routing. The concept of Areas basically introduced in MVC 2. So, basically it creates boundaries; to be more precise functional boundaries. This gives you flexibility to isolate large and

complex sections into different areas. This way it is easier to manage large enterprise project. Now, for using these areas we 1st need to register these areas. So, Area Routes are basically configured by creating classes for each area which derives from the "***AreaRegistration***" class by overriding **Area Name** and **Register Area**. So, if you open "***Global.asax***" file, you will find that there is call for registering all the areas during the start event.

Now, there could be scenario wherein user may end up having "***Area Conflicts***" scenario. This basically happens if there are two controllers in your project with the same name. You

might have registered one controller within your area and the other one in root of your application. But, what will happen if there is a call for the controller comes irrespective of its nature, it will create an ambiguity and hence it will land up in exception. In order to avoid this scenario, the easiest possible way is use unique name of the controllers, but if you can't avoid that, then you can provide a namespace which will be used for locating the controller like shown below in the snippet.

```
//Movie/Avatar
            routes.MapRoute("Movie",
                "Movie/{name}",
                new{controller="Movie",action="Search",name=UrlParameter.Optional},
                new[] {"AreaMovie.Controllers"});
```

Here, the fourth parameter which I have

specified is basically meant for array of namespaces. So, the controller for my movie project now lives in "***AreaMovie.Controllers***". Now, the next thing which we could see here is "***Catch-All Parameter***". Now, the Catch-All parameter allows for a route to match any of the parameters passed as query string. So, below is the snippet which we need to use for this

```
routes.MapRoute("catchAll", "yourQuery/{query-name}/{*params}");
```

And this will be declared in "***RegisterRoutes***" static method. So, this can match any of the incoming query with following patterns.

URL	Parameter Value
yourQuery/Query1/param1/param2	Params = "param1/ param2"
yourQuery/Query1/param1	Params = "param1"
yourQuery/Query1/	Params = "", still valid

Ignore Routes:-

We have also discussed this while designing our 1st route. But, let me explain the same in brief. There are certain requests which we can prevent by defining a route and using "***StopRoutingHandler***" class. So, basically when a request is handled by "***StopRoutingHandler***" class, the "***StopRoutingHandler***" class blocks any additional processing of the route. We use the same in order make sure that request don't map to a file on the disk

and you don't want routing to handle the same.

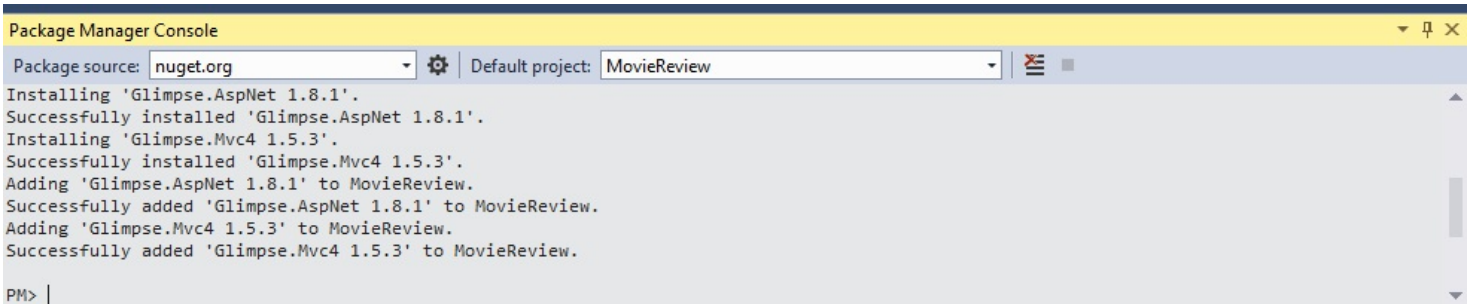
```
public static void RegisterRoutes(RouteCollection routes)
{
        routes.Ignore("{resource}.axd/{*pathInfo}");
        new StopRoutingHandler();
}
```

So, in the above snippet we are ignoring any request which is coming for a resource ending with "***.axd***" extension. Now, since the route is returning "***StopRoutingHandler***();", it will pass the request for normal ASP.Net processing, which in this case falls under normal HTTP Handler mapped to handle the "***.axd***" extension.

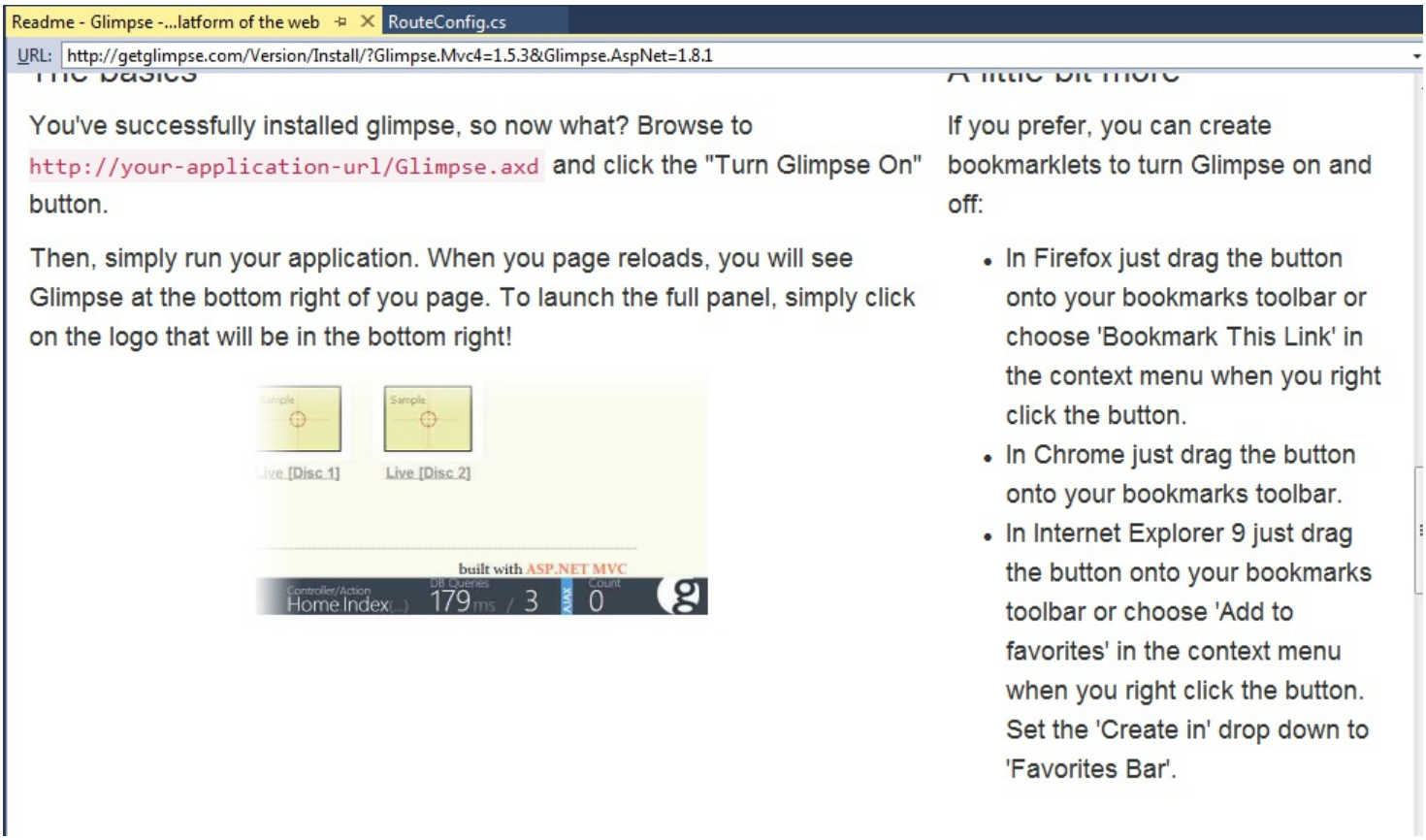
Debugging Routes:-

Now, this is the place which I enjoy most. I use "***Glimpse***" to debug my routes. Glimpse is like "***FireBug***" client side debugger; only difference here is it takes care of all server side codes, its **Network**, **Routes**, **Actions**, **Traffic** everything. So, I must say that Glimpse is one of the most important tools. Developer must have in his/her development box. For me, Actually Glimpse gives Clarity in order to understand routing better, it gives me exact picture like which was the controller invoked, what was the action, which routing principle picked up and many more thing. You can install

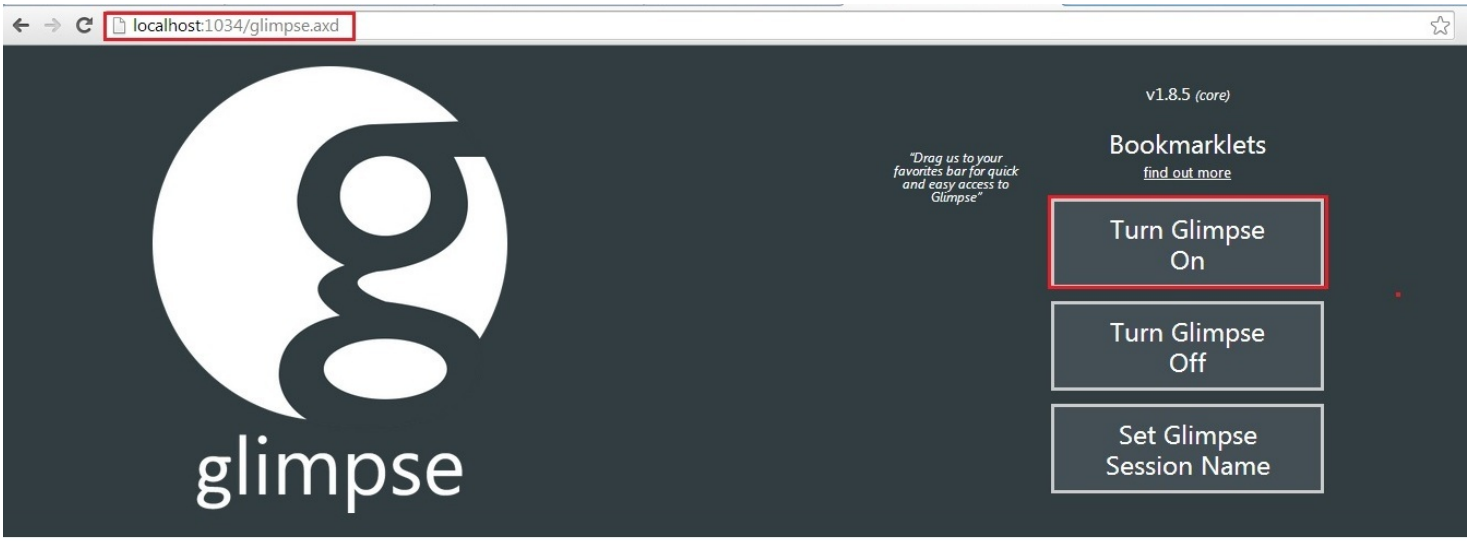
Glimpse both ways via "***Nuget or via Pacakage Manager Console Window***". So, you can install Glimpse via powershell command "*Install-Package Glimpse.Mvc4*" as I am using MVC 4 version for development. For different versions, different versions of glimpse are available.



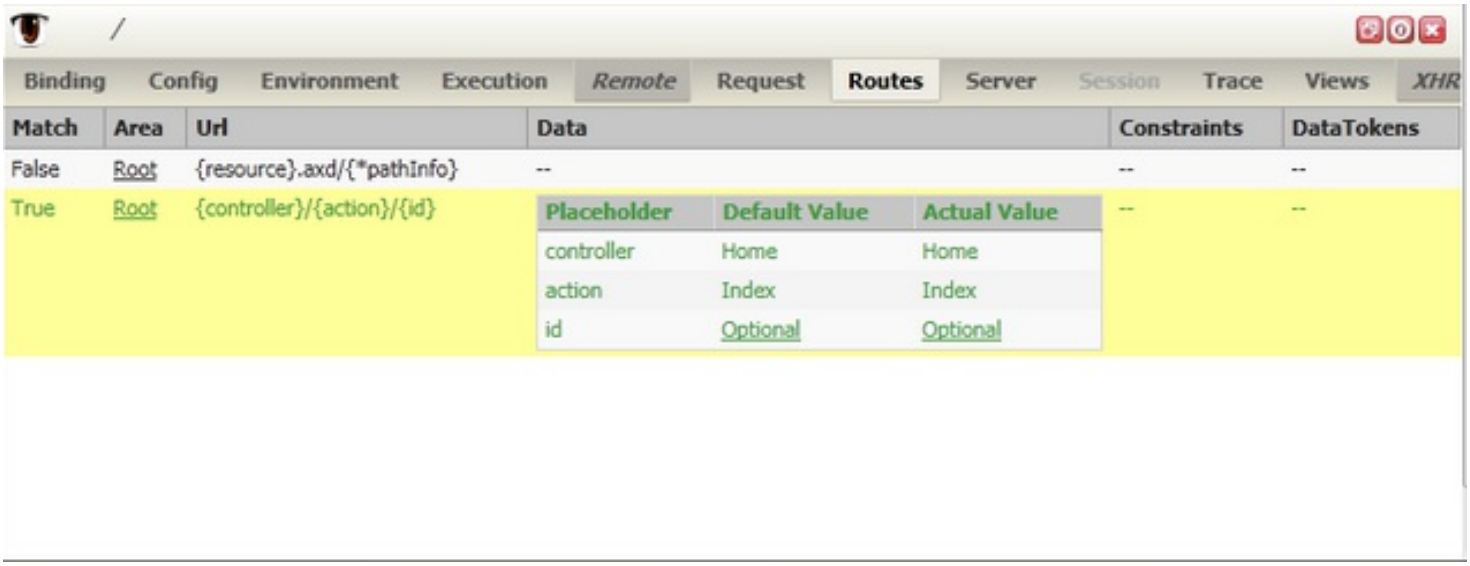
So, once successfully installed, it will open readme file in Visual Studio as shown in the below screen shot. It will have general instruction that how to enable Glimpse in your application.



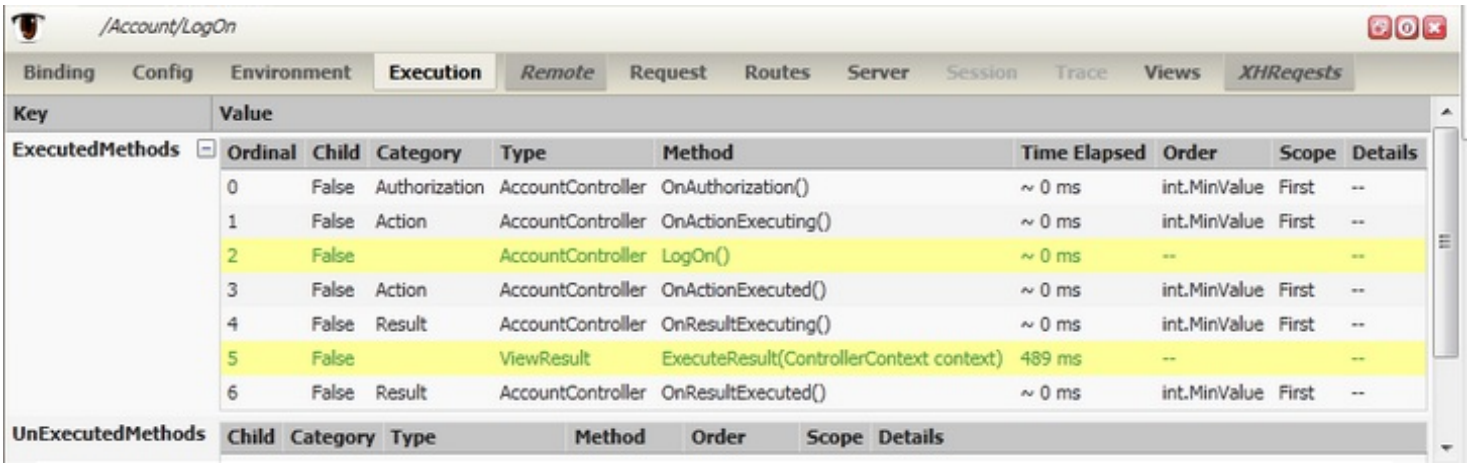
So, it basically says that you need to go into the root of your application and you need to type "***Glimpse.axd***" as shown below in the screen shot.



Upon typing that, it will open the above shown window, then you need to click on "***Turn Glimpse On***" button, and then refresh the home page, then one banner type tool will be popped up at the bottom of the screen as shown below.



So, as you can see in the above screen shot, it has all the relative sections for server side and currently, it's showing the Routes data. You can also use the same for execution path as shown below. So, currently it's showing the path for "***Account/LogOn***". It is giving the complete details in terms of which method in that controller has taken what time with categories.



So, I find this tool pretty useful in debugging and checking the path exactly.

Constructing URLs From Route:-

ASP.NET routing enables you to programmatically generate URLs that are based on route definitions. You specify parameter values and if it is required a route name in that, and ASP.NET generates a URL string that corresponding to a matching route. So, if you change route URL patterns, you do not have to change the code or markup that generates URLs unless the parameters in the patterns are changed. Basically, When ASP.NET routing constructs a URL; it uses the First route which has a pattern that matches the

parameters which you provide, unless you specify a route name. If you specify a route name, ASP.NET uses the named route.

Below are the steps for generating the URL from the Route:-

Create an instance of the "***RouteValueDictionary***" class that contains all the parameter values that you want to include in the URL. The parameters must match a route URL pattern. Then, the "***RouteValueDictionary***" object can include parameters in addition to the ones which match a route URL pattern. If it does this, those values are assigned to

the query string.

Then, Call the "***GetVirtualPath***" method of the "***RouteCollection***" class and pass the "***RouteValueDictionary***" object in order to construct the URL. To indicate a specific route, include the route name. "***The GetVirtualPath***" method returns a "***VirtualPathData***" object.

And last but not the least from the "***VirtualPathData***" object; retrieve the "***VirtualPath***" property to get the string that represents the URL.

Now, below is the sample code explaining the same.

```
RouteValueDictionary parameters =
   new RouteValueDictionary
      {
         { "Releaseyear", "2014" } ,
         { "category", "movie" }
      };

VirtualPathData obj =
   RouteTable.Routes.GetVirtualPath(null, "MoviesRoute", parameters);

HyperLink1.NavigateUrl = obj.VirtualPath;
```

So, what is happening here I have created "***RouteValueDictionary***" with parameters which I have passed to the "***VirtualPathData***". Then, I have assigned the same to the hyperlink. Now, this way I am creating the URL, from the code based on the route parameters. So, here is the thing that routes work two ways for incoming URLs, it maps the request and for outgoing it constructs the

URL. So, this is the basic difference between routing and URL Re-writing. URL Re-writing only maps the request.

Route Data:-

Now, let's understand what is Route Data? So, route data basically contains information about the route that matches the request. So, when a request for" ***/Movies/Details/1***" comes in, the route attempts to match the request. If it matches, it then creates a dictionary that contains information parsed from the URL. Specifically, it adds a key to the dictionary for each URL parameter in the route URL. In the case of

"***{controller}/{action}/{id}***", the dictionary will contain at least three keys: "***controller***," "***action***," and "***id***. " In the case of" ***/Movies/Details/1***", the URL is parsed to supply values for these dictionary keys. In this case, "***controller = Movies***", "***action = Details***", and "***id = 1***".

So, the "***RouteData***" is the property of the "***RequestContext***" that is used in MVC to keep the route values.

Route Constraints:-

Now, in addition to matching a URL request to a route definition by the

number of parameters in the URL, we can also specify these values in the parameters which meet certain constraints. So, if a URL contains values that are outside the constraints for a route, that route is not used to handle the request. You can then add constraints to make sure that the URL parameters contain values that will work in your application. So, now constraints are defined by using regular expressions or by using instance which implement "***IRouteConstraint***" interface. So, when you add the route definition to the Routes collection, you can then add constraints by creating a "***RouteValueDictionary***" object that contains the verification test. The key in

the dictionary identifies the parameter that the constraint applies to. The value in the dictionary can be either a string that represents a regular expression or an object that implements the "***IRouteConstraint***" interface. Now, a fairly good example in this case could be referred from the url "***http://msdn.microsoft.com/en-us/library/vstudio/cc668201(v=vs.100).c***

Summary:-

So, in this chapter we have seen couple of important bullet points like what are MVC Areas. How this is extremely useful in isolating the complex functional

scenarios; how to register them. Then, we have seen how to skip routing for any physical file lying on the disk. Then, we have seen how to use Glimpse to debug the routes. We have also covered URL generation from the route. In the end we have seen Route Data and constraints.

Chapter 12: Web API and Dependency Injection

WHAT DO you find in this CHAPTER?

⇒ Introduction

⇒ Why Web API?

⇒ Creating a new empty Project

⇒ Adding Web API to an ASP.Net Project

⇒ Creating your 1st controller

⇒ Dependency Injection

⇒ Implementing POST

⇒ Implementing DELETE

⇒ Implementing PUT

⇒ Summary

Introduction:-

So, whenever we are building APIs on the web, there are several ways to do that. These include:-

HTTP RPC: - So, it means using HTTP in Remote Procedure call to call methods across the web. These verbs could be "***GetMovies, InsertMovies, DeleteMovie***" etc, and that each of these will have different endpoints with different URIs in there. So, this is one of the models which we used to follow.

Restful: - This is another technique, which is also popular. It also includes the idea of URI endpoints which again point to functionality, but here we are using Resource URIs instead. So, we are using the same in such a way that let's suppose we are going against a movie endpoint like "***Movie***" and that is using "***Get***" HTTP Verb to retrieve data and again for inserting data doing "***Post***" against the endpoint. So, it also includes the concept of "***Stateless Server***" and "***Content-Negotiation***". Hence, here the caller can specify that what kind of data is interested in. So, there is no compulsion of single server, so it can better scaled up.

HATEOS: - It stands for "**Hypermedia as the engine of Application State**". Hmm quite a bit long acronym but idea here is to extend the REST-based interfaces to include things like relationships which are called as hypermedia. So, the content which we are going to receive is again what we have been getting with the earlier methods, like "***GetMovie***". So, here may be a link to this which can enable you to Add a new movie, Fetch a movie and many more things.

Why Web API:-

So, before jumping to answering this

question; let's go back and think for a minute that what options we had earlier. They are:-

WCF: - This is one of the famous concepts of building web service. So, here we can take the existing SOAP-based stack which exposes and allow the same in a way that JSON Objects, or non-SOAP Objects to be returned from these services. So, WCF is all about configuration. But, sometimes this could be a tedious job to do.

ASP.Net MVC: - This is the one which could be used to extend the controller model to return the JSON data to accept HTTP Verbs like Get, Post, Put or

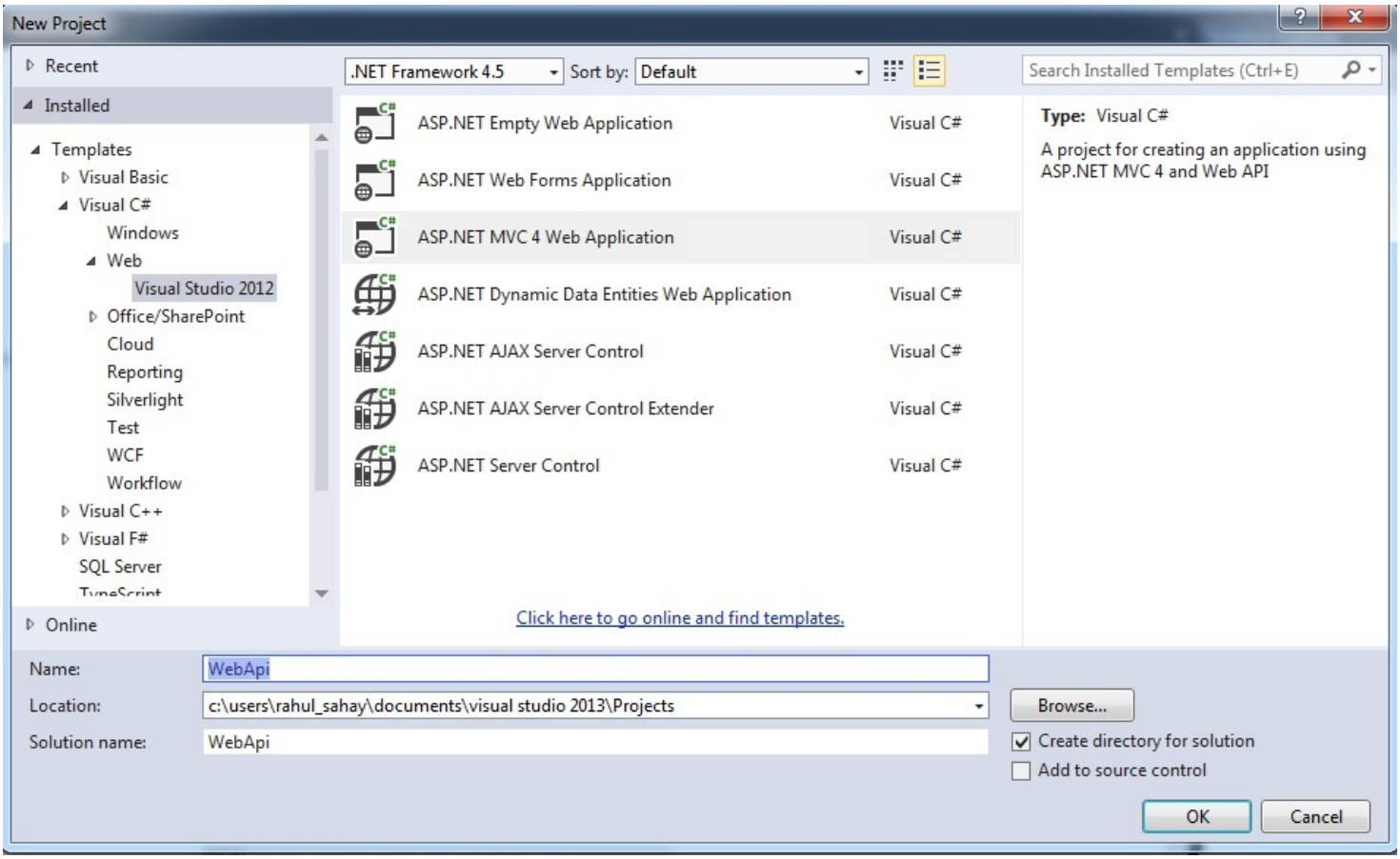
Delete. This may be a difficult scenario as we often try to match the name of the verb inside our action method. And in this scenario, there could heavy chances that it would lead to name conflict with same parameters even though you are having different HTTP Verbs.

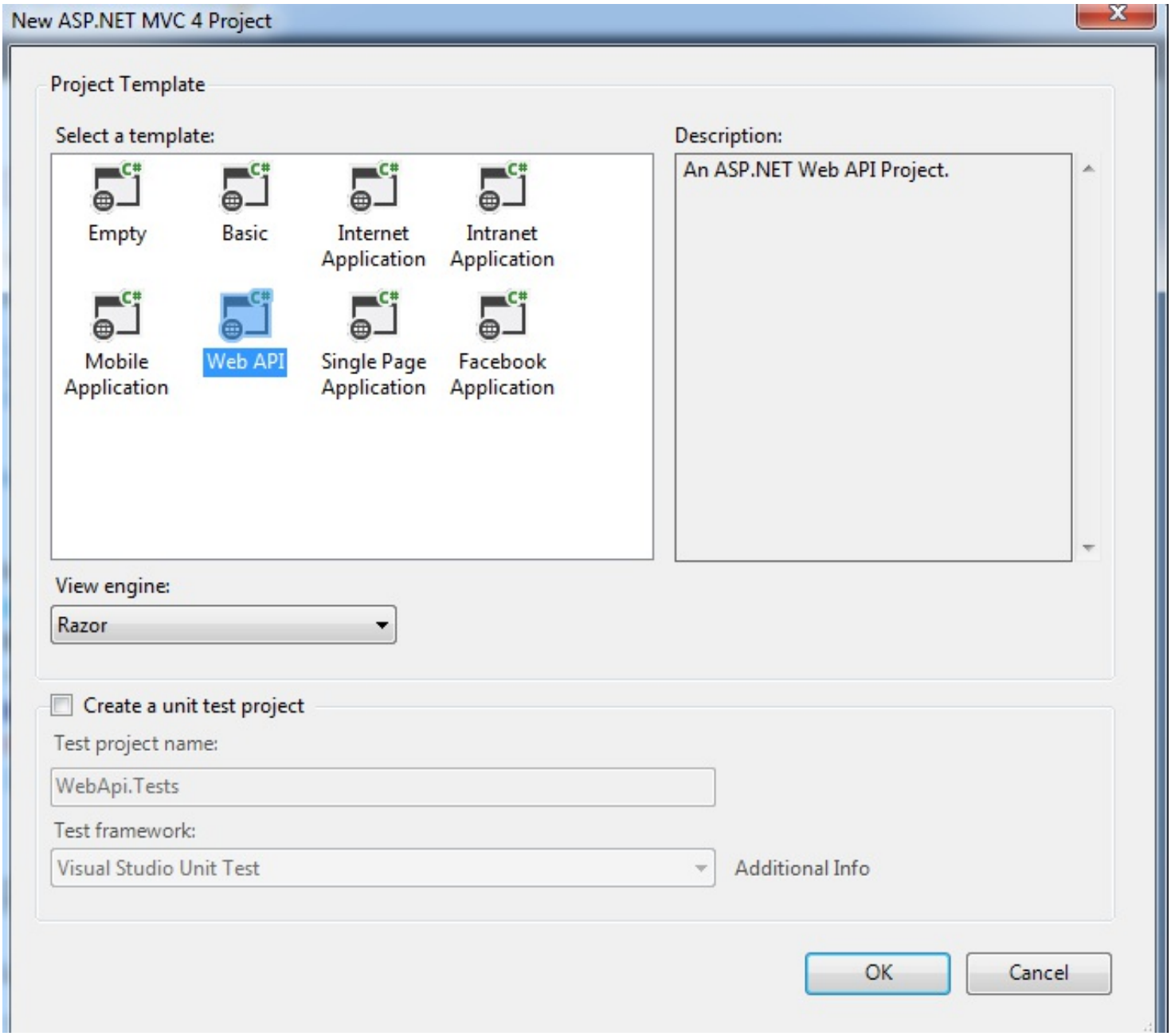
ASP.Net Web API: - This one is easier to implement because HTTP Verbs are directly mapped to these methods. So, any method with the name "***Get***" actually makes a "***Get***" request and similarly other HTTP Verbs do what they mean. And, as it takes clearer and crisp convention for exposing each of the controllers that are specific to REST-based APIs. Now, it also encourages

separation of what we are mapping and what we are designing. So, it means we can have controllers more self-describing which means we can get the idea by just looking at the controllers. Now, as far as hosting is concerned, it can also be hosted outside "***IIS***". But, one prerequisite here is ASP.Net Web API does require .NET 4.0 or above.

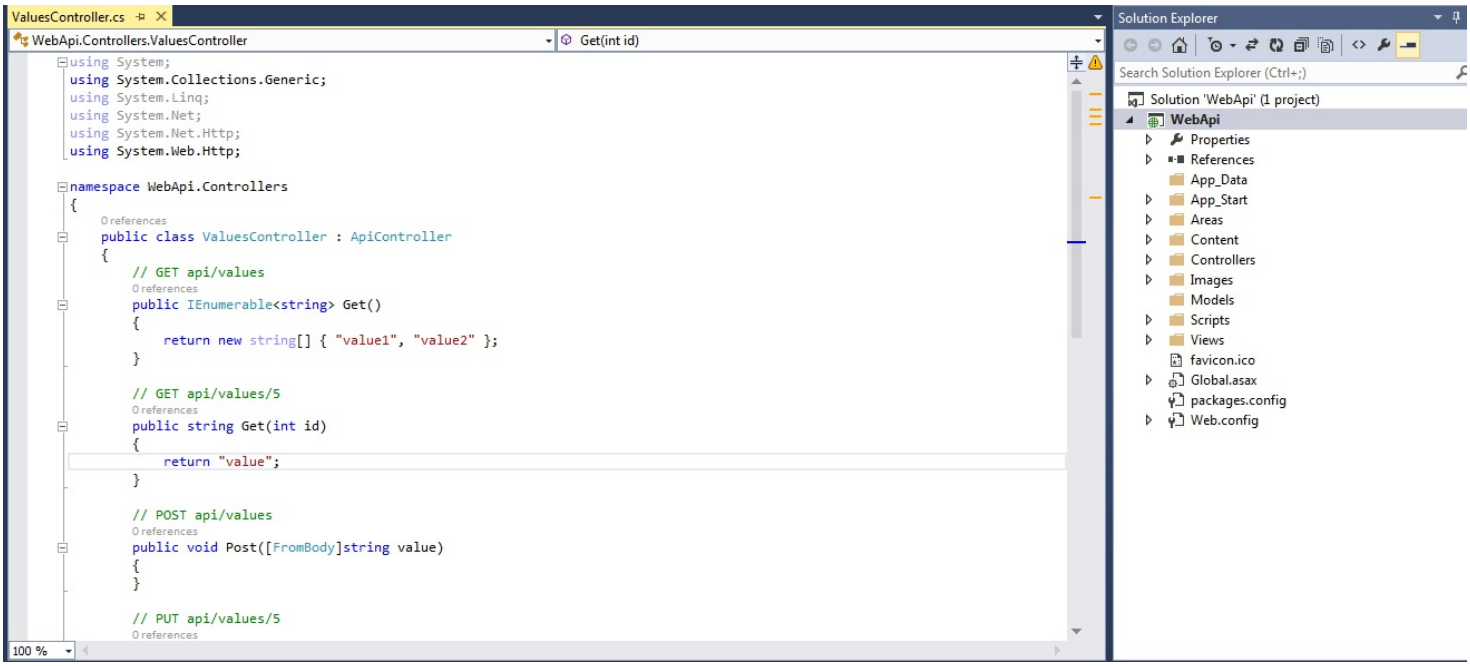
Creating a new Project:-

So, I'll begin by adding one New MVC 4 web project as shown below in the screen shot.

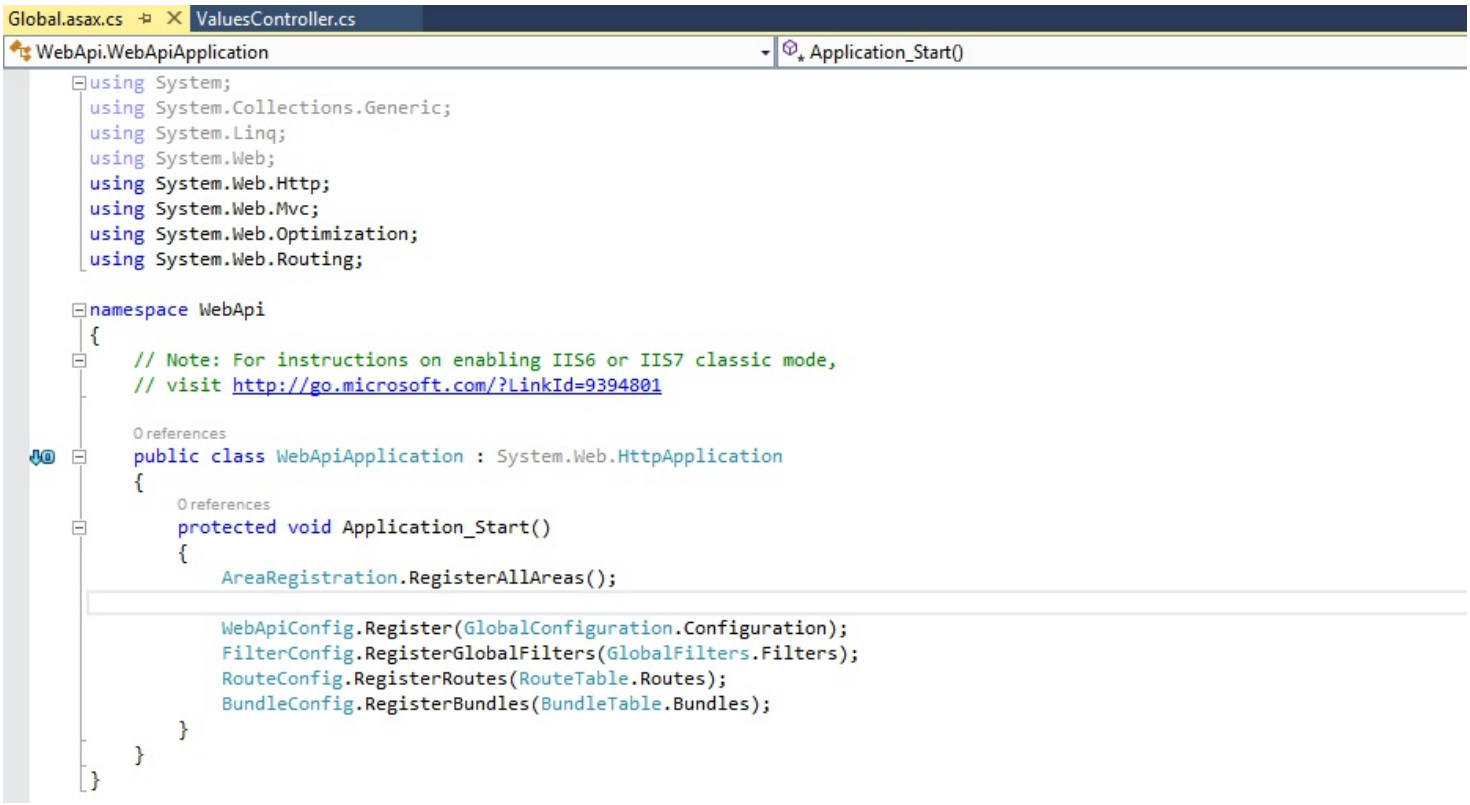

New Project
Recent
Installed
Templates
Visual Basic
Visual C#
Windows
Web
Visual Studio 2012
Office/SharePoint
Cloud
Reporting
Silverlight
Test
WCF
Workflow
Visual C++
Visual F#
SQL Server
Online
.NET Framework 4.5
Sort by:
Default
Search Installed Templates (Ctrl+E)
ASP.NET Empty Web Application
Visual C#
ASP.NET Web Forms Application
Visual C#
ASP.NET MVC 4 Web Application
Visual C#
ASP.NET Dynamic Data Entities Web Application
Visual C#
ASP.NET AJAX Server Control
Visual C#
ASP.NET AJAX Server Control Extender
Visual C#
ASP.NET Server Control
Visual C#
Type: Visual C#
A project for creating an application using ASP.NET MVC 4 and Web API
Click here to go online and find templates.
Name:
WebApi
Location:
c:\users\rahul_sahay\documents\visual studio 2013\Projects
Browse...
Solution name:
WebApi
Create directory for solution
Add to source control
OK
Cancel



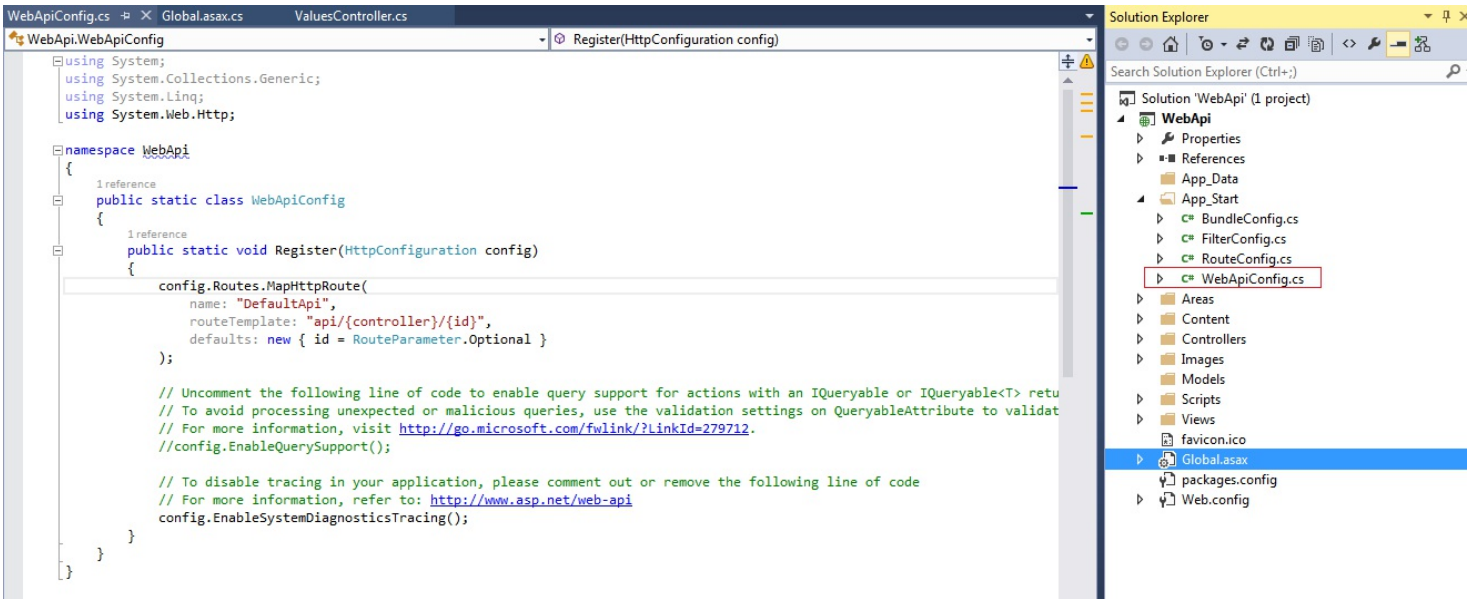
Now, here I'll choose a Web API project. Now, below is the screen shot of sample Web API project in its default state.



Now, if you see the global.asax file, you will see that, there's an "***Application_Start***" section.

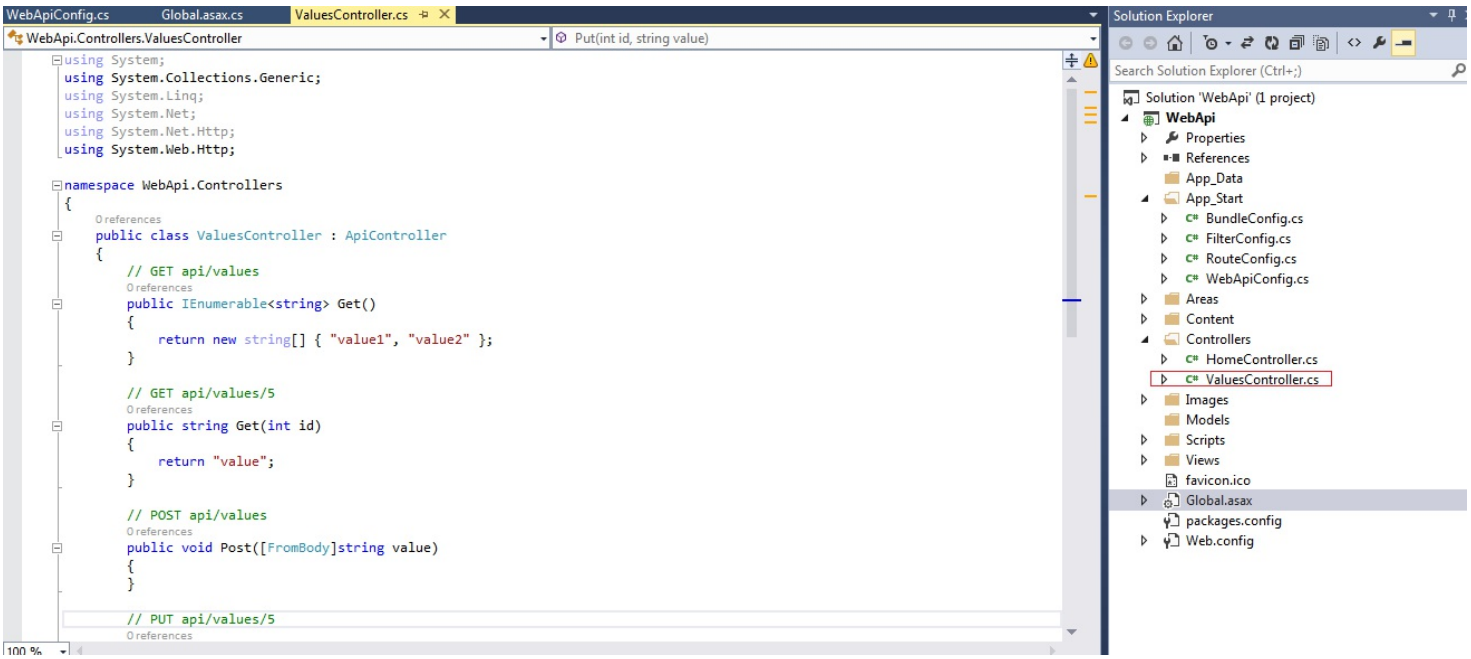


In this section, all the configurations take place. So, we can see here there is a call for registering "***WebAPIConfig***". When you do "***F12***" on this call, you will land up in a class which sits in "***App_Start***" folder.

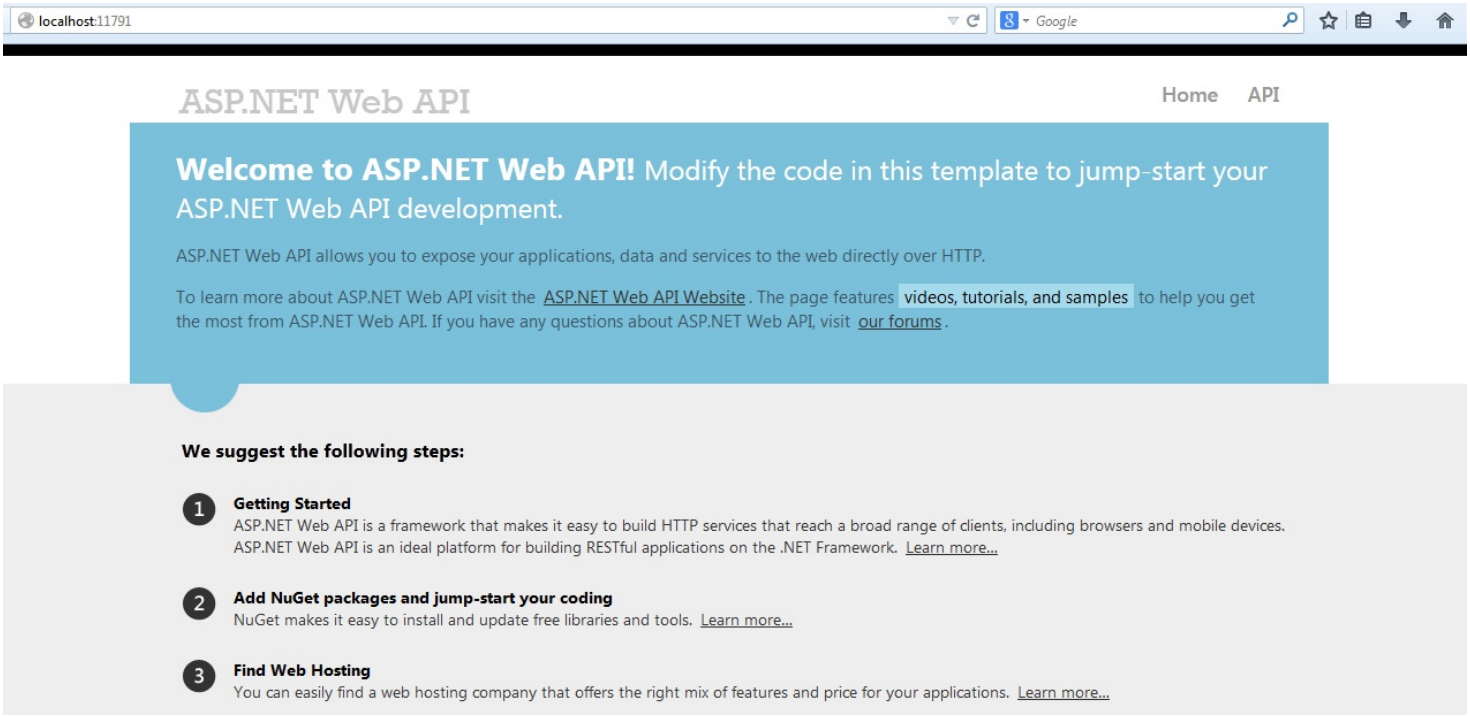


As you can see that there is one method which will take care of startup configuration. The very 1st thing it does is configure default route for Web API. So, when it hits the server with a URI, it will try to resolve this against these routes. Now, it has also created me a controller's folder. So, when you open the same, you will see "***ValuesController***" in there. This is pretty much similar to MVC controller.

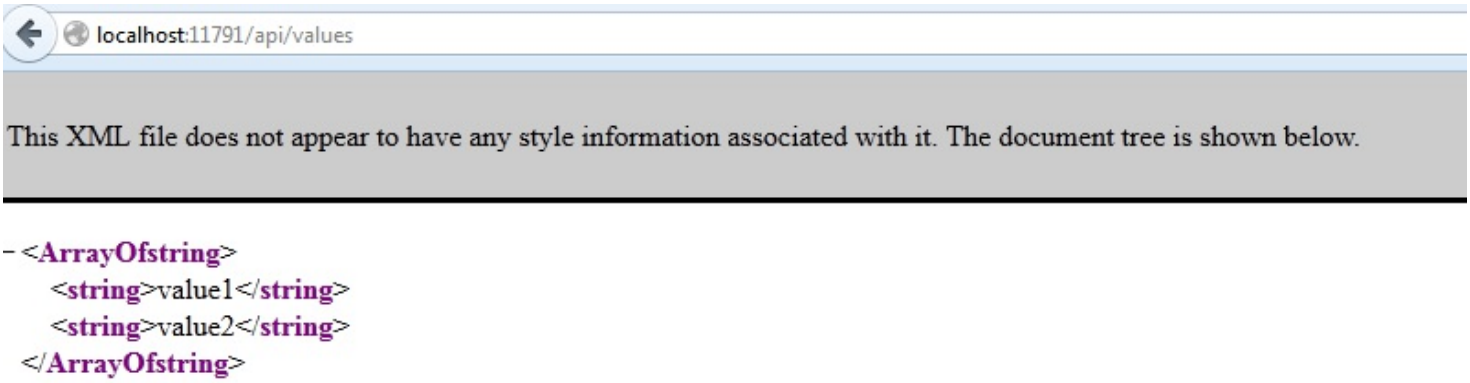
But, the basic difference here is that it gets inherited from "***ApiController***"



So, this "***ApiController***" knows that we are going to do this HTTP based mapping. So, here if a request comes for "**api/Values**", it is going to look for values controller. Now, let's go ahead and run the application for a minute. Then, it will produce the below output.



Now, if go ahead in the address bar and type the URL "***http://localhost:11791/api/values***" , then it will produce me a "***Get***" result but in xml format.



But, I want the same in JSON format, now in order to get the same in JSON format I need to configure the settings in my "***WebApiConfig***". I need to specify below two lines of code in front of the existing code as shown below.

```
namespace WebApi
{
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            //Below formatter is used for returning the JSON result.
           Var appXmlType =
               config.Formatters.XmlFormatter.SupportedMediaTypes.FirstOrDefault(t
=> t.MediaType == "application/xml");
```

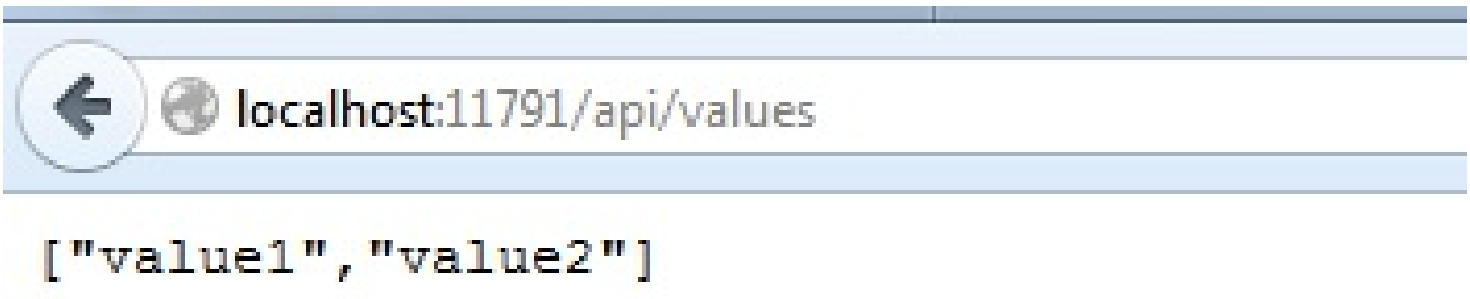
```
            config.Formatters.XmlFormatter.SupportedMediaTypes.Remove(appXmlType);

            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );

            // Uncomment the following line of code to enable query support for
actions with an IQueryable or IQueryable<T> return type.
            // To avoid processing unexpected or malicious queries, use the
validation settings on QueryableAttribute to validate incoming queries.
            // For more information, visit
http://go.microsoft.com/fwlink/?LinkId=279712.
            //config.EnableQuerySupport();

            // To disable tracing in your application, please comment out or
remove the following line of code
            // For more information, refer to: http://www.asp.net/web-api
            config.EnableSystemDiagnosticsTracing();
        }
    }
}
```

Now, when you do build and run then it will produce the exact output in JSON Format as shown below.



So, if you go back to values controller and look at the "***Get***" method; then you will come to know that; this is the method which got invoked without any additional parameter. Now, let me go ahead and try to change the values in this method and see whether it's taking the effect or not. So, as you can see below in the snippet, I have provided my name there as value, and same has been returned as JSON.

```
// GET api/values
        public IEnumerable<string> Get()
        {
            return new string[] { "Rahul", "Sahay" };
        }
```



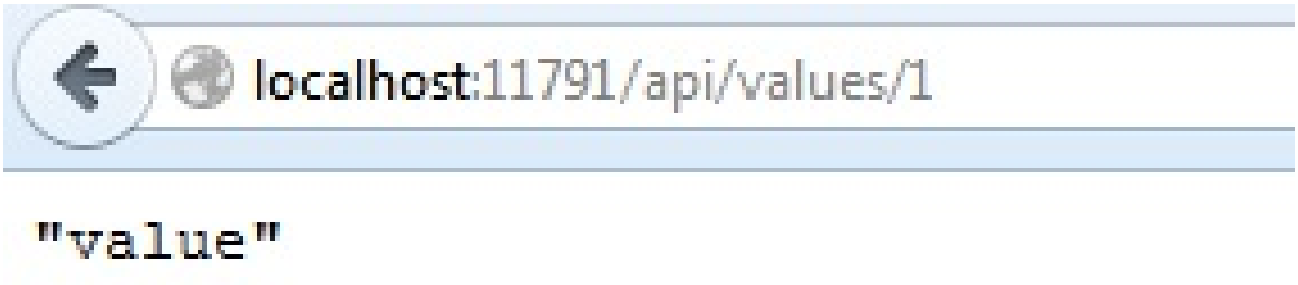

```
["Rahul","Sahay"]
```

Now, if I go ahead and look the different "***Get***" method, it takes one id as shown below.

```
// GET api/values/5
        public string Get(int id)
        {
            return "value";
        }
```

In our Web API route pattern this Id is already specified. But, Id specified after the controller is specified as optional parameter. So, if I go ahead and invoke

this URL "***http://localhost:11791/api/values/1***", then it will return the below output.



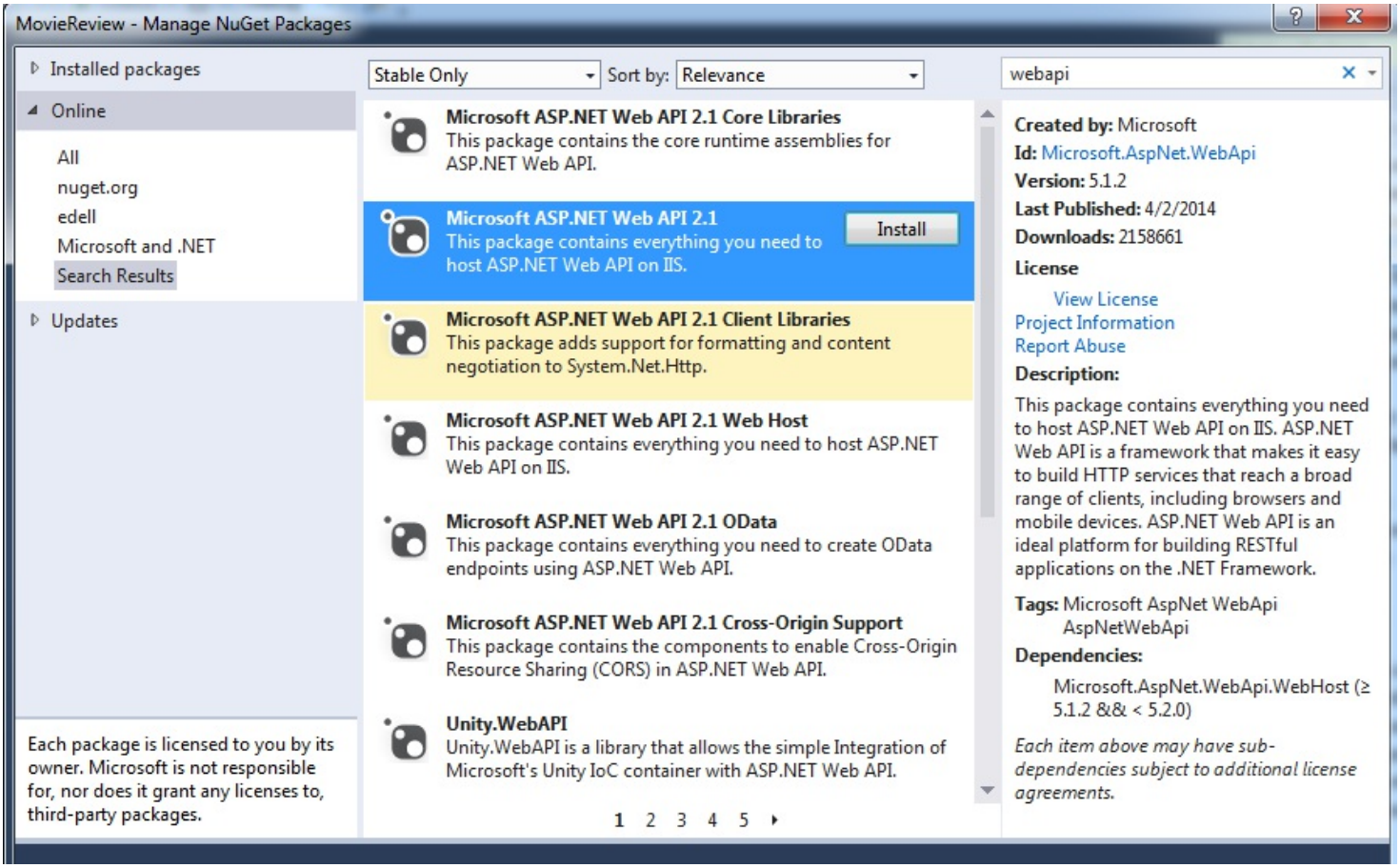
So, these are all basically routing mechanism, how you have configured it in the Web API route structure.

Adding Web API to an existing Project:-

Now, in this case, I am going to add Web

API in my existing project which we have already built as "***MovieReview***".

So, in this project, 1st am going to add Web API reference in the project using the nuget package manager as shown below.



Now, once it is successfully installed,

you can see in the global.asax file, call for registering Web API has been added there.

```
namespace MovieReview
{
    // Note: For instructions on enabling IIS6 or IIS7 classic mode,
    // visit http://go.microsoft.com/?LinkId=9394801

    0 references
    public class MvcApplication : System.Web.HttpApplication
    {
        0 references
        protected void Application_Start()
        {

            var migrator = new DbMigrator(new Configuration());
            migrator.Update();

            if (!WebSecurity.Initialized)
            {
                WebSecurity.InitializeDatabaseConnection("DefaultConnection", "UserProfile", "UserId", "UserName", autoCreateTables: true);
            }
            AreaRegistration.RegisterAllAreas();

            WebApiConfig.Register(GlobalConfiguration.Configuration);
            FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            BundleConfig.RegisterBundles(BundleTable.Bundles);
            AuthConfig.RegisterAuth();
        }
    }
}
```

Now, again when you do "***F12***" on register, then it will take you the config file.

```
namespace MovieReview
{
    1 reference
    public static class WebApiConfig
    {
        1 reference
        public static void Register(HttpConfiguration config)
        {

           //Below formatter is used for returning the Json result.
            var appXmlType = config.Formatters.XmlFormatter.SupportedMediaTypes.FirstOrDefault(t => t.MediaType == "application/xml");
            config.Formatters.XmlFormatter.SupportedMediaTypes.Remove(appXmlType);

            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );

            // Uncomment the following line of code to enable query support for actions with an IQueryable or IQueryable<T> return type.
            // To avoid processing unexpected or malicious queries, use the validation settings on QueryableAttribute to validate incoming queries.
            // For more information, visit http://go.microsoft.com/fwlink/?LinkId=279712.
            //config.EnableQuerySupport();
        }
    }
}
```

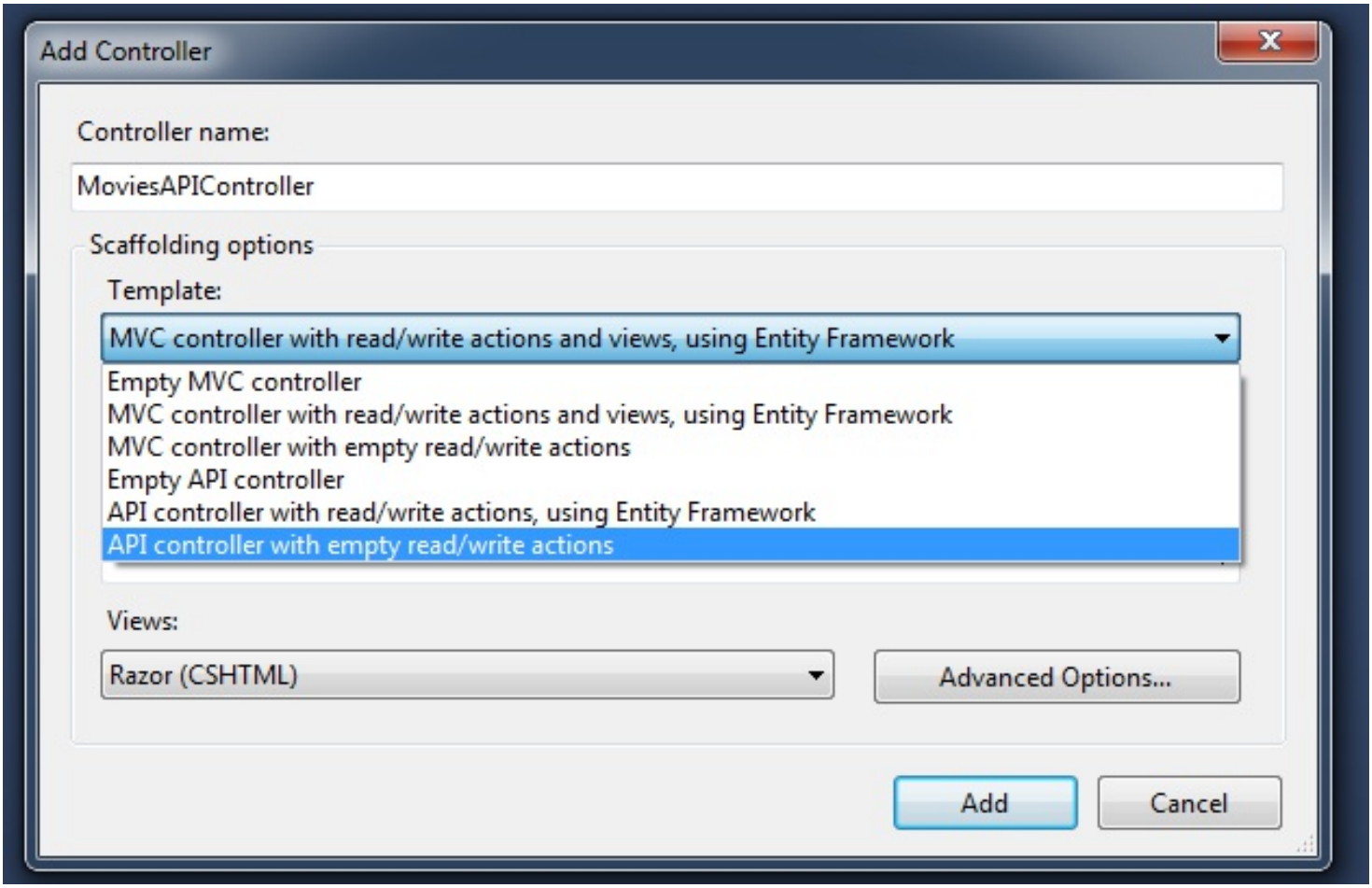
Now, if you can see in the above screen shot, by default query support is commented. So, basically this is an optional support which you can get in WEB API to expose an "***IQueryable***" input. So, the benefit of this could be that it allows you to expose your data as "***OData***" feeds. Now, by definition "**OData** is a standardized protocol for creating and consuming data APIs. OData builds on core protocols like HTTP and commonly accepted

methodologies like REST. The result is a uniform way to expose full-featured data APIs.” But, I am leaving this currently commented as we are not working currently with OData. This is extremely useful in case you want to support for feeds. Now, if you want to delve more, you can refer ***http://www.odata.org/*** site for the same.

Creating your 1st controller:-

So, now here I am going to create my 1st Web API controller. Now, I can have a different sub folder inside my

controller's folder for "***Web-API***" or I can have altogether different folder dedicated for WEB API, whichever way you are comfortable that's fine. But for now I am going to create the new controller in the controller's folder itself. So, what you need to do is right-click on the controllers' folder as then add "***Web-API***" controller from the drop down as shown below.



So, as you can see in the screenshot, last three options in the dropdown are dedicated for API controllers. Now, from these options, I usually prefer "***Empty API Controller***" and then I write my required operations in there. So, now I'll create a Get method which will get the instance of my "***MovieDb***".

In the later section of this course we'll see how to simplify this instance creation process. But, for now that's ok, and I always believe that you start small, test that thing, then apply all sort of design patterns to it. So, now, I have a Get method and in here I'll get some data as shown below.

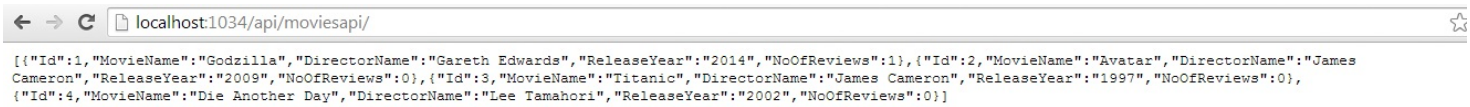
```
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using MovieReview.Models;

namespace MovieReview.Controllers
{
    0 references
    public class MoviesAPIController : ApiController
    {
        private IMovieDb _movieDb;
        0 references
        public object Get()
        {
            _movieDb = new MovieDb();
            var results =
                _movieDb.Query<Movie>().OrderByDescending(m => m.reviews.Count())
                    .Select(m => new MovieViewModel
                    {
                        Id = m.Id,
                        MovieName = m.MovieName,
                        DirectorName = m.DirectorName,
                        ReleaseYear = m.ReleaseYear,
                        NoOfReviews = m.reviews.Count()
                    });
            return results;
        }

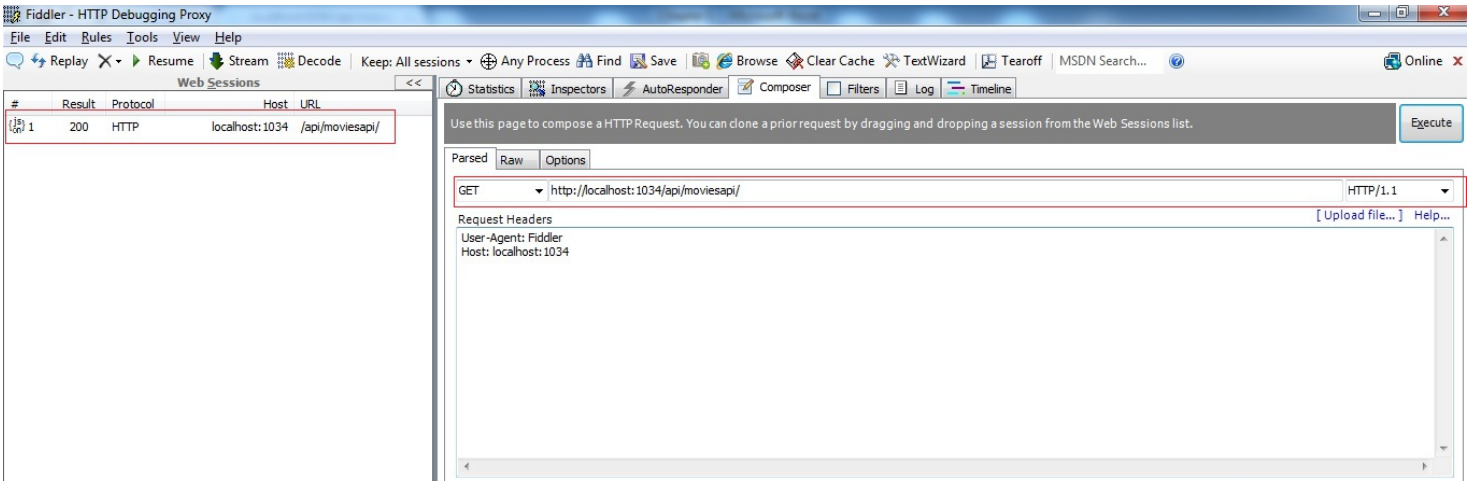
    }
}
```

Here as you can see in the above screen shot, I have just used the same query, what I have used earlier in Home controller. So, now when I run the app and navigate to URL ***http://localhost:1034/api/moviesapi/***. It

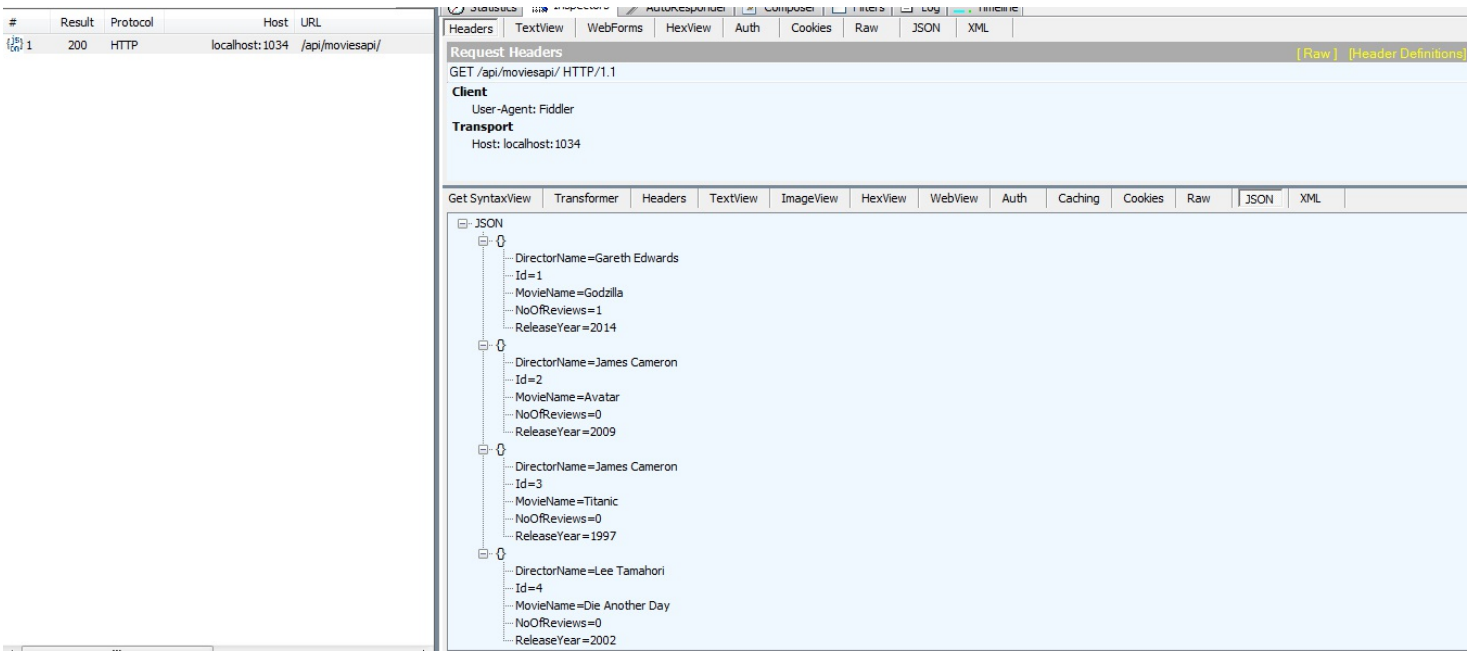
will produce me the below output in JSON format.



Here, you can also verify the same in "***Fiddler***". Fiddler is a very good tool in order to capture the network traffic; here we just want to make sure what requests have been processed, whether it's processed correctly or not. So, Fiddler is basically a must have tool for any web developer. You can get the same online. So, you can see in the below screen shot, I have composed a "***Get***" request from the Fiddler, and it returned me a JSON object with "***200 HTTP Status***".



Now, when you click on this returned object highlighted in the left pane, you can see the data returned from the server.



Dependency Injection:-

Now, let's talk about Dependency Injection. So, this is the stuff which is very popular these days in developers' arena for obvious reasons. So, let's discuss few of them. Now, actually dependency injection is well applied with repository pattern wherein I am using one repository to get hold on my database and do all "***CRUD***" operation for me. So, if you can see the code I am creating a reference of my movie Db. Now, here I would like to have a way to take these services and supply them to controllers. Because in future there may be as many services as required, so I need a way to simplify these for me and dependency injection is the answer to achieve it.

So, here I'll be creating one constructor with required interface in there. In this case this would be "***MovieDb***" Interface. So, if you would have been using "***Repository Pattern***", then in that case, this interface would be for that Repository which you would like to use. Now, the reason for using these interfaces enables you to write tests easily against these controllers. Now, here I am going to use new local variable "***_moviedb***" inside my constructor

```
namespace MovieReview.Controllers
{
    1 reference
    public class MoviesAPIController : ApiController
    {
        private IMovieDb _movieDb;

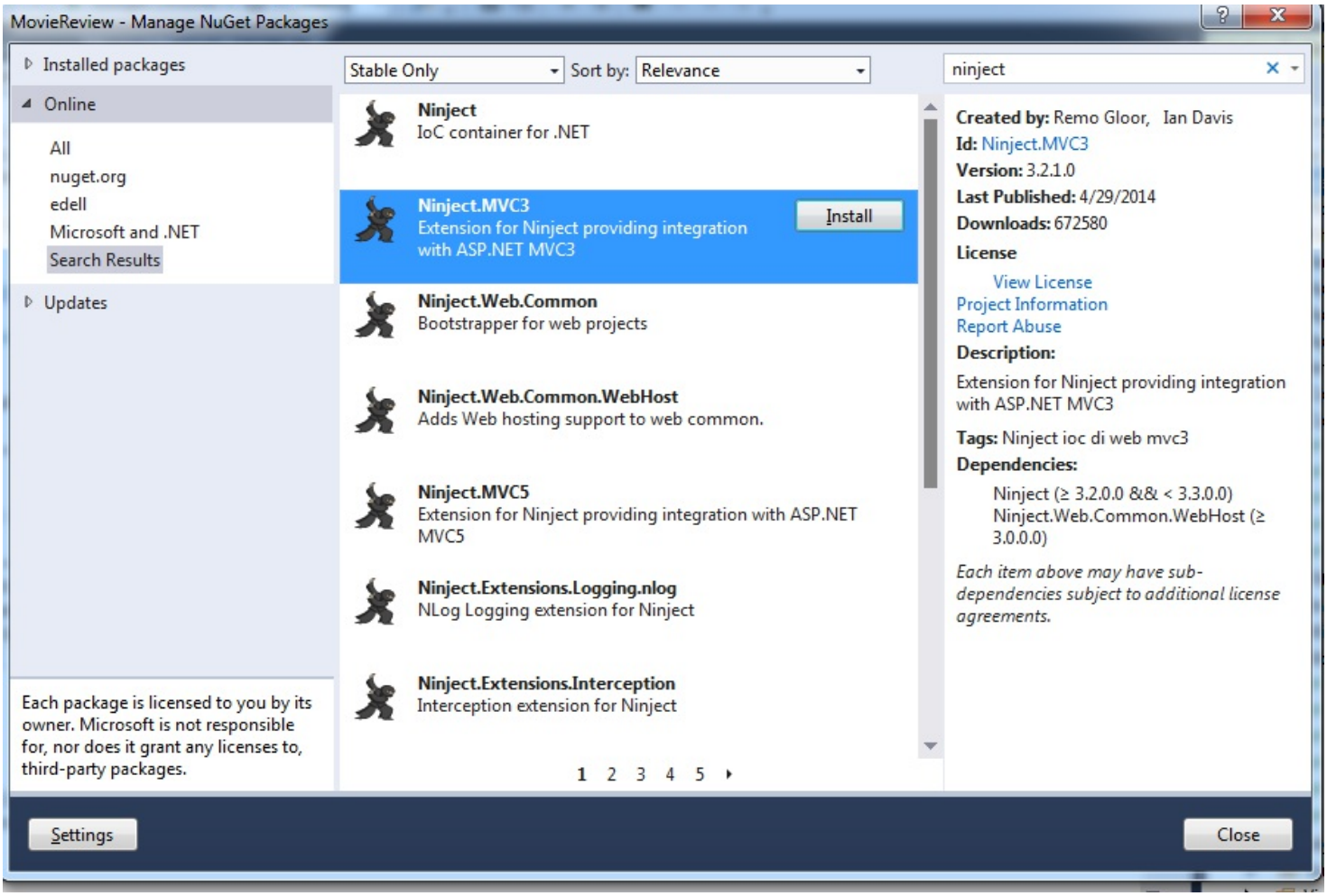
        0 references
        public MoviesAPIController(IMovieDb movieDb)
        {
            _movieDb = movieDb;
        }

        0 references
        public object Get()
        {
            //   _movieDb = new MovieDb();
            var results =
                _movieDb.Query<Movie>().OrderByDescending(m => m.reviews.Count())
                    .Select(m => new MovieViewModel
                    {
                        Id = m.Id,
                        MovieName = m.MovieName,
                        DirectorName = m.DirectorName,
                        ReleaseYear = m.ReleaseYear,
                        NoOfReviews = m.reviews.Count()
                    });
            return results;
        }
    }
```

Now, if you build the above piece of code and try invoking API in the browser, then this time, it will result an error message as shown below.



And the error here says that controller doesn't have the "***parameterless public constructor***". Now, the defaults in the Web API are that all of these controllers must not have any parameter in it. So, we are going to change this behavior by allowing dependency injection container. Hence, here the technique which I am going to use is "***Ninject***". However, you could use any of your favorites like "***StructureMap, Unity***" etc. So, the job of "***Ninject***" over here is to pass in these dependencies into a controller as required. Now, for doing the same, we 1st need to install one package, i.e. is "***Ninject.MVC3***". Now, don't go by naming here, it applies to MVC 4 and above as well.



Now, once successfully installed, this is going to add one new class in the "***App_Start***" folder with the name "***NinjectWebCommon.cs***". So, if you look at this class you will get the idea that this is startup bootstrap class. So, when the web server comes and gets started, it's going to do its own

registration. Now, below is the snippet for this class.

```
[assembly:
WebActivatorEx.PreApplicationStartMethod(typeof(MovieReview.App_Start.NinjectWebCo
mmon), "Start")]
[assembly:
WebActivatorEx.ApplicationShutdownMethodAttribute(typeof(MovieReview.App_Start.Nin
jectWebCommon), "Stop")]

namespace MovieReview.App_Start
{
    using System;
    using System.Web;

    using Microsoft.Web.Infrastructure.DynamicModuleHelper;

    using Ninject;
    using Ninject.Web.Common;
```

```
    public static class NinjectWebCommon
    {
        private static readonly Bootstrapper bootstrapper = new Bootstrapper();

        /// <summary>
        /// Starts the application
        /// </summary>
        public static void Start()
        {
            DynamicModuleUtility.RegisterModule(typeof(OnePerRequestHttpModule));
            DynamicModuleUtility.RegisterModule(typeof(NinjectHttpModule));
            bootstrapper.Initialize(CreateKernel);
        }

        /// <summary>
        /// Stops the application.
        /// </summary>
        public static void Stop()
        {
            bootstrapper.ShutDown();
        }

        /// <summary>
        /// Creates the kernel that will manage your application.
        /// </summary>
        /// <returns>The created kernel.</returns>
        private static IKernel CreateKernel()
        {
            var kernel = new StandardKernel();
            try
            {

kernel.Bind<Func<IKernel>>().ToMethod(ctx=>()=>newBootstrapper().Kernel);

kernel.Bind<IHttpModule>().To<HttpApplicationInitializationHttpModule>();

                RegisterServices(kernel);
                return kernel;
            }
            catch
            {
                kernel.Dispose();
                throw;
            }
        }
```

```
        /// <summary>
        /// Load your modules or register your services here!
        /// </summary>
        /// <param name="kernel">The kernel.</param>
        private static void RegisterServices(IKernel kernel)
        {
        }
    }
}
```

Here, at the bottom you can see that there's a static method "***RegisterServices***". So, this is the place where you are actually going to add the code to register the type of services. Now, here we can do the same by using "***kernel***" object and then we say "***Bind***" and pass in the interface which we are interested in. So, in this case it would be "***IMovieDb***" and then bind it to "***MovieDb***". So, it's actually telling here,

when someone request this; then give them the instance of this service. But, here again I need to register my context as well, because my Interface "***IMovieDb***" has interdependency on this. So, I need to resolve this as well. Hence, the modified code looks like below.

```
using MovieReview.Models;

[assembly:
WebActivatorEx.PreApplicationStartMethod(typeof(MovieReview.App_Start.NinjectWebCo
mmon), "Start")]
[assembly:
WebActivatorEx.ApplicationShutdownMethodAttribute(typeof(MovieReview.App_Start.Nin
jectWebCommon), "Stop")]

namespace MovieReview.App_Start
{
    using System;
    using System.Web;

    using Microsoft.Web.Infrastructure.DynamicModuleHelper;

    using Ninject;
    using Ninject.Web.Common;

    public static class NinjectWebCommon
    {
        private static readonly Bootstrapper bootstrapper = new Bootstrapper();

        /// <summary>
        /// Starts the application
        /// </summary>
```

```
        public static void Start()
        {
            DynamicModuleUtility.RegisterModule(typeof(OnePerRequestHttpModule));
            DynamicModuleUtility.RegisterModule(typeof(NinjectHttpModule));
            bootstrapper.Initialize(CreateKernel);
        }

        /// <summary>
        /// Stops the application.
        /// </summary>
        public static void Stop()
        {
            bootstrapper.ShutDown();
        }

        /// <summary>
        /// Creates the kernel that will manage your application.
        /// </summary>
        /// <returns>The created kernel.</returns>
        private static IKernel CreateKernel()
        {
            var kernel = new StandardKernel();
            try
            {

kernel.Bind<Func<IKernel>>().ToMethod(ctx=>()=>newBootstrapper().Kernel);

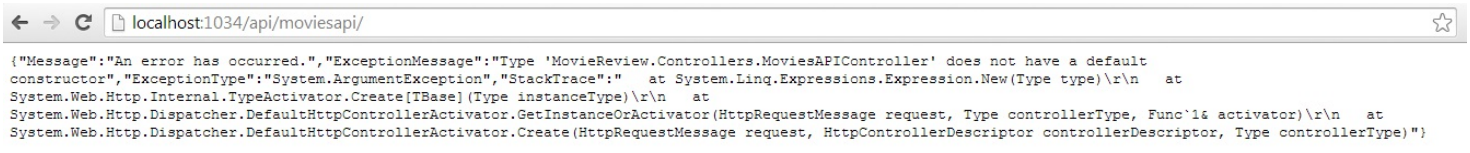
kernel.Bind<IHttpModule>().To<HttpApplicationInitializationHttpModule>();

                RegisterServices(kernel);
                return kernel;
            }
            catch
            {
                kernel.Dispose();
                throw;
            }
        }

        /// <summary>
        /// Load your modules or register your services here!
        /// </summary>
        /// <param name="kernel">The kernel.</param>
        private static void RegisterServices(IKernel kernel)
        {
```

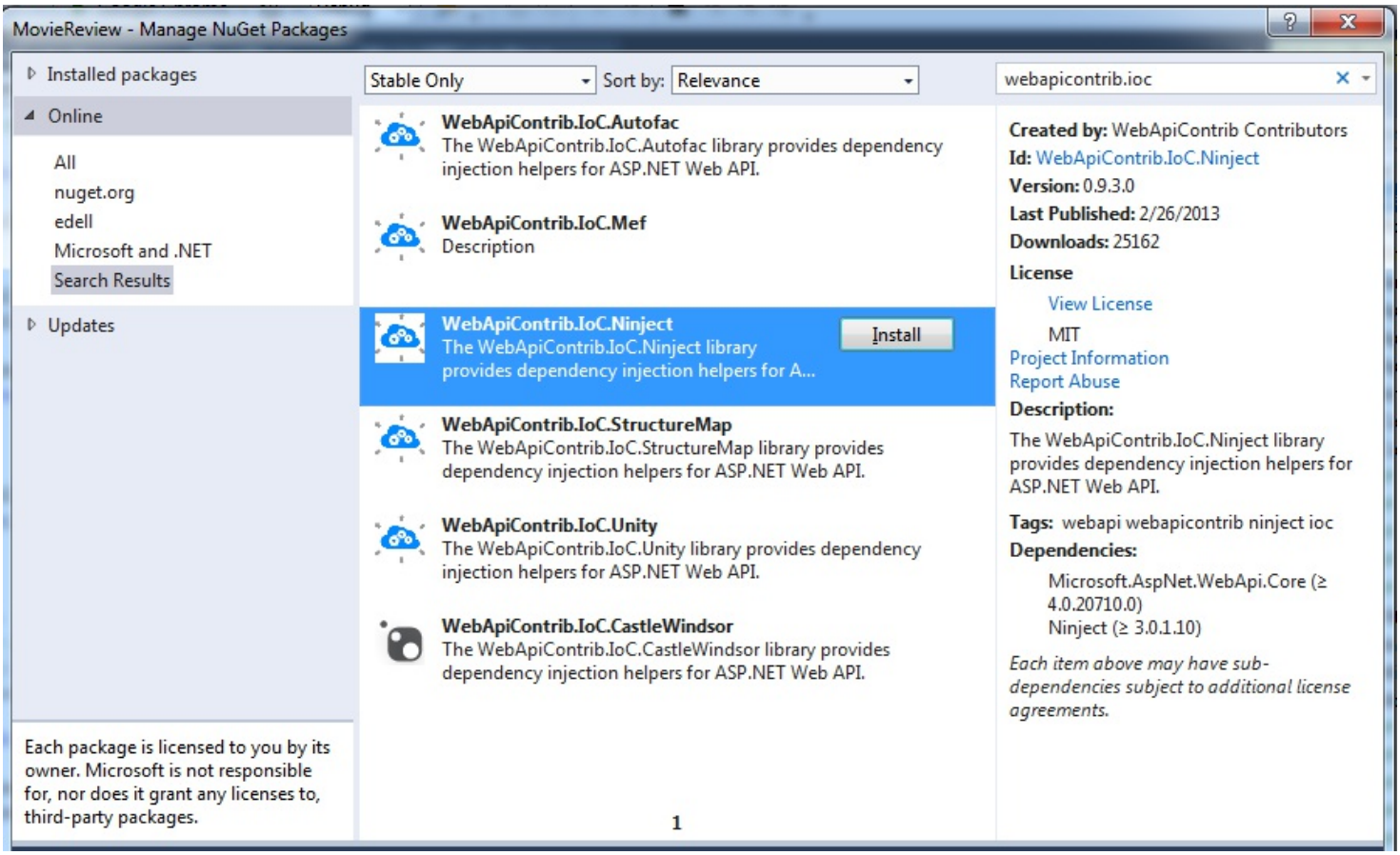
```
            kernel.Bind<IMovieDb>().To<MovieDb>();
            kernel.Bind<MovieDb>().To<MovieDb>();
        }
    }
}
```

Now, if I go ahead and build again, I'm going to get the same error as shown below.



So, now why I get the same error. Now, I have Dependency Injection as well to solve my dependencies. So, to answer this question, "***DI***" doesn't know about Web API by default. Hence, to fix the same, I need to go ahead and install one

more package as shown below.

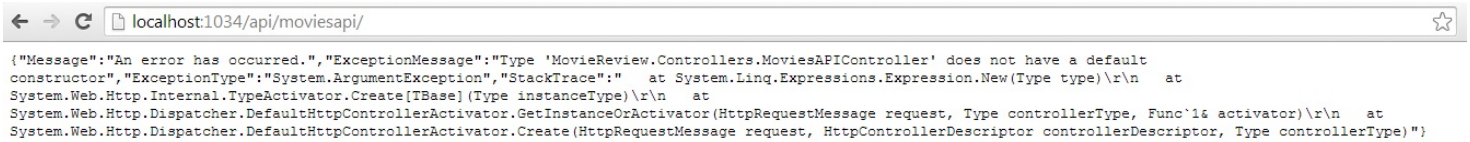


Now, this utility is going to help you to build the Web API components and expose the same against Dependency Injection as well. So, now in the "***NinjectWebCommon***", I need to add settings to support Web API as shown below.

```
private static IKernel CreateKernel()
        {
            var kernel = new StandardKernel();
            try
            {
kernel.Bind<Func<IKernel>>().ToMethod(ctx=>()=>newBootstrapper().Kernel);
kernel.Bind<IHttpModule>().To<HttpApplicationInitializationHttpModule>();
                // Web API settings
GlobalConfiguration.Configuration.DependencyResolver=newNinjectResolver(kernel);
                RegisterServices(kernel);
                return kernel;
            }
            catch
            {
                kernel.Dispose();
                throw;
            }
        }
```

Now, with the above piece of code in place, when I go ahead build the solution and refresh the page, it will give the required output. But, this time via Dependency Injection. So, basically here we have implemented "***Get***" via

“***Dependency Injection***”.



Implementing POST:-

So, in case of POST, I will be using attribute as “***[FromBody]***” in my Post method. It's basically an attribute which you can put directly on parameter. And this is going to tell Web API, that we want to pass into the Post an object model. So, here we would like to infer the JSON into an instance of Movie Model from the body of the Post. So, here my code will look like as shown

below

```
public object Post([FromBody] Movie movie)
        {
            try
            {

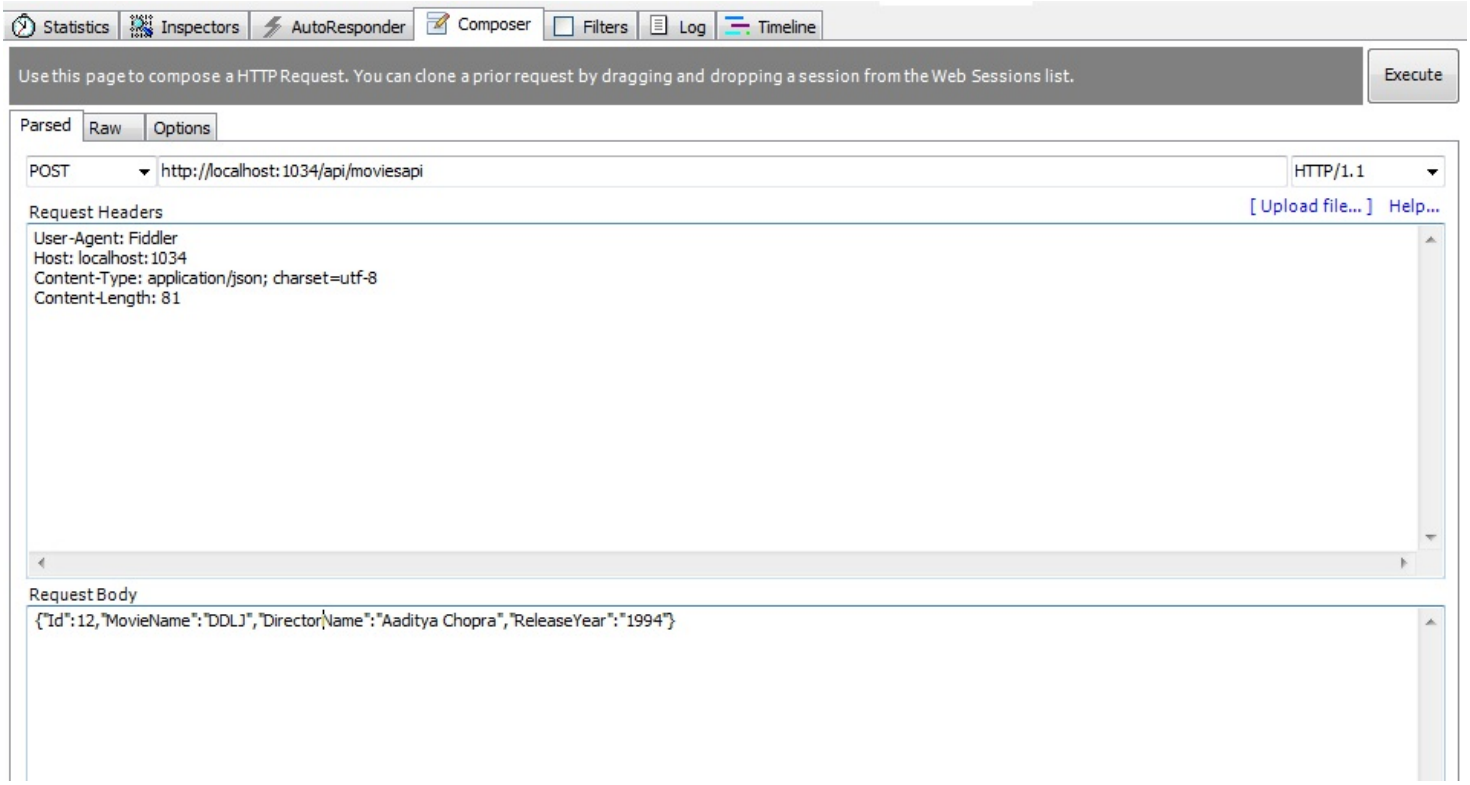
                if (ModelState.IsValid)
                {
                    var context = new MovieDb();

                    //Check Duplicate
                   if (_movieDb.Query<Movie>().Any(m => m.Id == movie.Id))
                    {
                        return
Request.CreateErrorResponse(HttpStatusCode.BadRequest, "Entry with the Id already
exist");
                    }
                    context.Movies.Add(movie);
                    context.SaveChanges();
                }
            }
            catch (Exception ex)
            {
                return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ex);
            }
            return Request.CreateResponse(HttpStatusCode.Created);
        }
```

Now, if you can see in the above code, I have also used couple of things, like I

took one instance for my context class just to commit the changes, then I also make sure that I won't be able to commit duplicate values to the db, Then if everything goes well, then I am just returning the HTTP Status code created, else Bad Request.

Here, let me explain the same using fiddler. So, as you can see below in the screen shot, I have composed a POST request in the fiddler, and when I click on Execute button, and then it will come in my debugger.




```
public object Post([FromBody] Movie movie)
{
    try
    {
        if (ModelState.IsValid)
        {
            var context = new MovieDb();

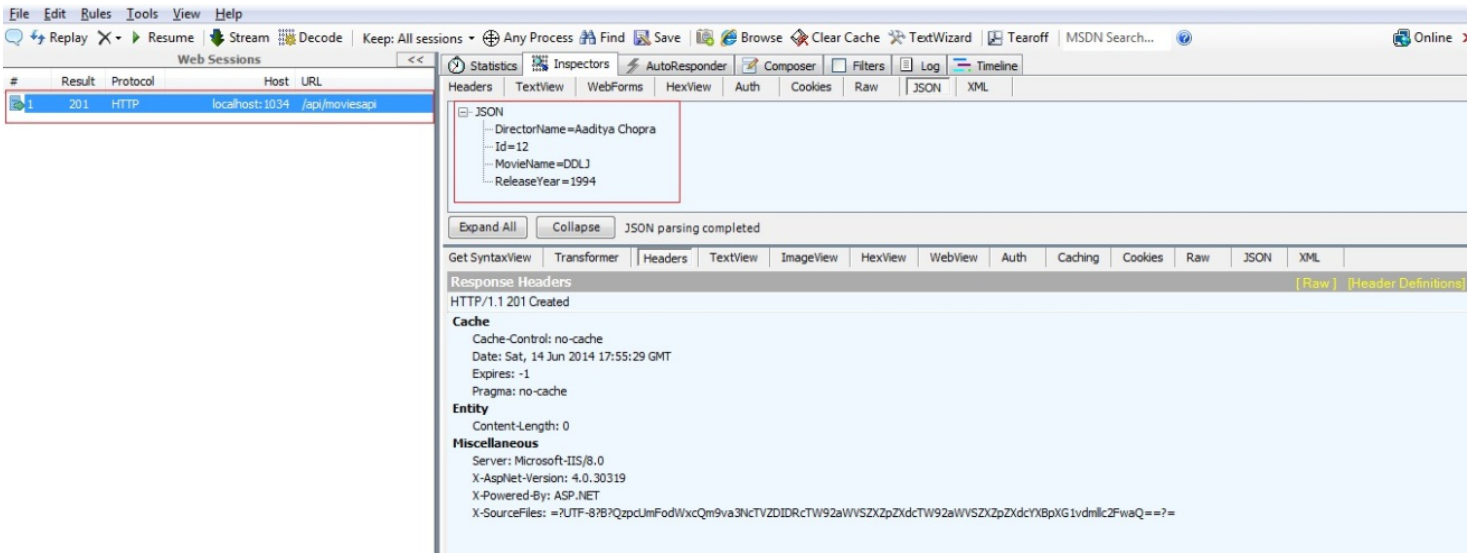
            //Check Duplicate
            if (_movieDb.Query<Movie>().Any(m => m.Id == movie.Id))
            {
                return Request.CreateErrorResponse(HttpStatusCode.BadRequest, "Entry with the Id already exist");
            }
            context.Movies.Add(movie);
            context.SaveChanges();
        }
    }
    catch (Exception ex)
    {
        return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ex);
    }
    return Request.CreateResponse(HttpStatusCode.Created);
}
```

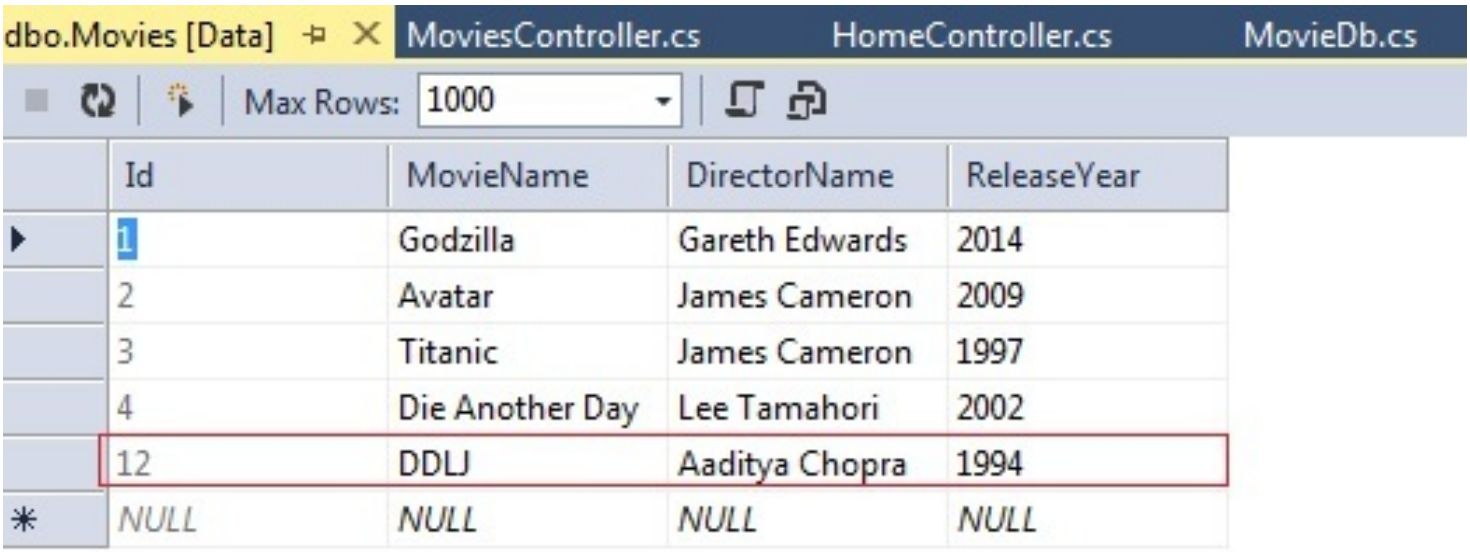
movie {MovieReview.Models.Movie}

DirectorName	"Aaditya Chopra"
Id	12
MovieName	"DDLJ"
ReleaseYear	"1994"
reviews	null

So, as you can see in the debugger, it came with following values, which is passed from the fiddler, and reviews is

coming null, because that collection I didn't pass. Now, once everything goes well, it will commit the same with following values and same will be reflected in db as shown below.



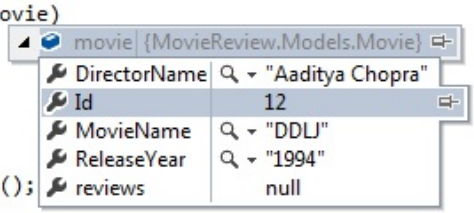


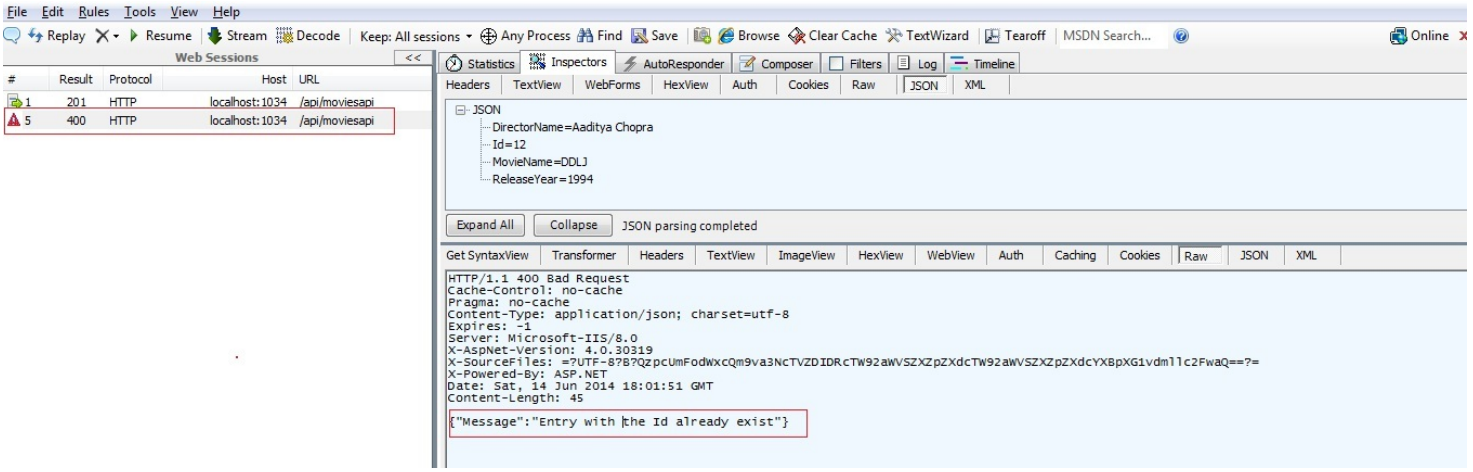
Id	MovieName	DirectorName	ReleaseYear
1	Godzilla	Gareth Edwards	2014
2	Avatar	James Cameron	2009
3	Titanic	James Cameron	1997
4	Die Another Day	Lee Tamahori	2002
12	DDLJ	Aaditya Chopra	1994
NULL	NULL	NULL	NULL

And in application; it will also reflect. Now, let me try re-execute the same Post body, this time I am trying to put the values with the same id, then in this case it should throw me an exception as shown below.

```
public object Post([FromBody] Movie movie)
{
    try
    {
        if (ModelState.IsValid)
        {
            var context = new MovieDb();

            //Check Duplicate
           if (_movieDb.Query<Movie>().Any(m => m.Id == movie.Id))
            {
                return Request.CreateErrorResponse(HttpStatusCode.BadRequest, "Entry with the Id already exist");
            }
            context.Movies.Add(movie);
            context.SaveChanges();
        }
    }
    catch (Exception ex)
    {
        return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ex);
    }
    return Request.CreateResponse(HttpStatusCode.Created);
}
```





So, this is how POST method works, now let's go ahead see how Delete works.

Implementing DELETE:-

Now, in this section, we are going to implement Delete method. So, below is the snippet of Delete implementation, and again this implementation is fairly simple. So, here I have passed Id as parameter to the method. So, now based

on this parameter it will go ahead and delete the record from the database.

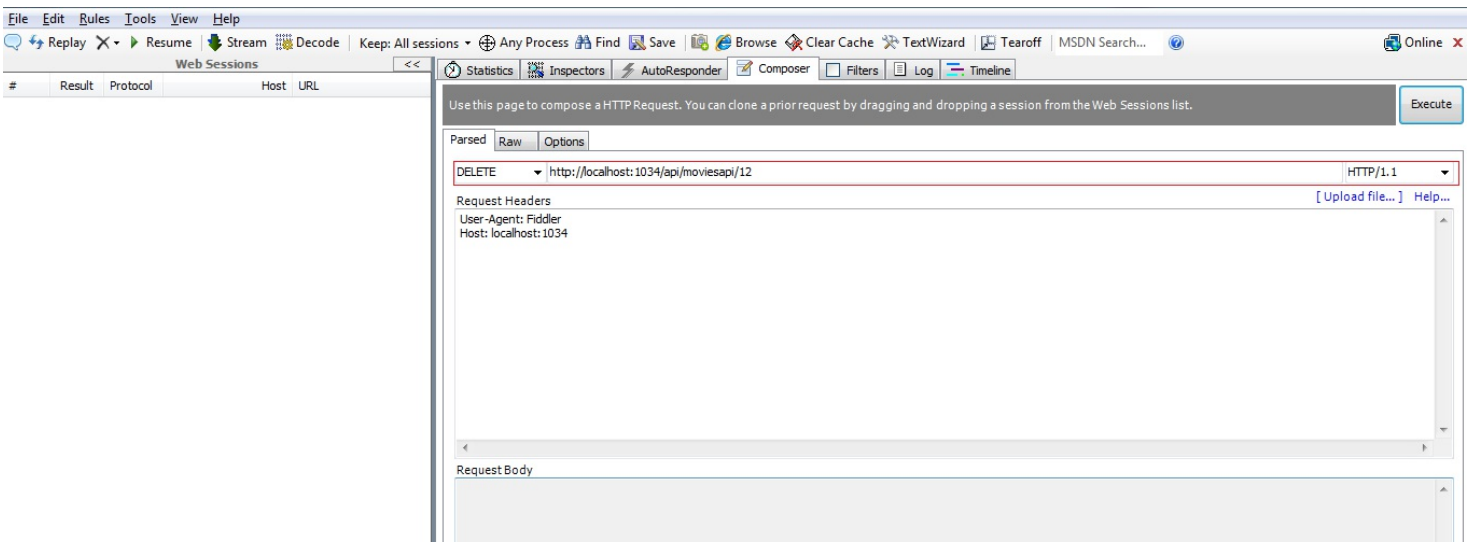
```
public HttpResponseMessage Delete(int Id)
        {
            try
            {
                if (ModelState.IsValid)
                {
                    var context = new MovieDb();

                    //If Id Not found
                    if (_movieDb.Query<Movie>().Any(m => m.Id == Id) == false)
                    {
                        Return Request.CreateResponse(HttpStatusCode.NotFound,
"Record Not found with the ID supplied");
                    }

                    //else delete
                    if (_movieDb.Query<Movie>().Any(m => m.Id == Id))
                    {
                        Movie movie = context.Movies.Find(Id);
                        context.Movies.Remove(movie);
                        context.SaveChanges();


                }
            }
        }
        catch (Exception ex)
        {
            return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ex);
        }
        return Request.CreateResponse(HttpStatusCode.OK);
}
```

Here, again in the 1st section of this code, it is trying to delete the record with the supplied id, but if record not found, then it will simply return **Record Not Found Message**. Otherwise, it will go ahead and delete the record from the database. Now, let's see the same in action. So, if you have noticed here, I have changed the verb from the dropdown and also passed id as parameter in the URL.



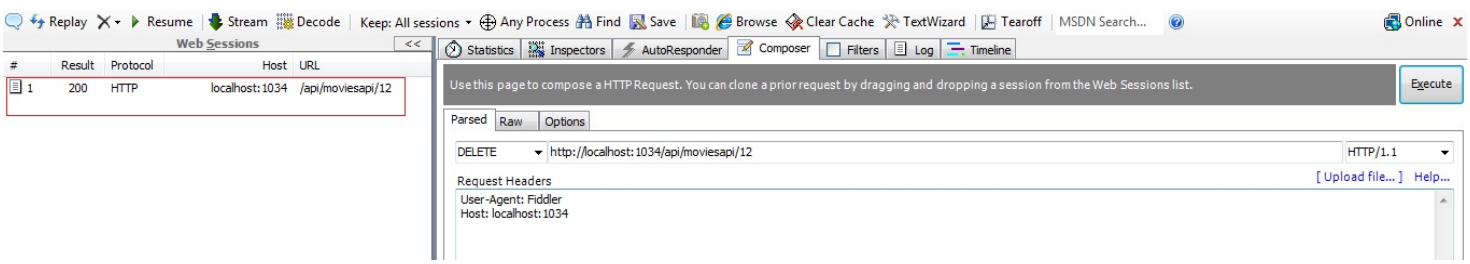
Now, when I click on Execute, then it will go ahead delete the desired record from the database as shown below.

```
0 references
public HttpResponseMessage Delete(int Id)
{                                          Id 12
    try
    {
        if (ModelState.IsValid)
        {
            var context = new MovieDb();

            //If Id Not found
            if (_movieDb.Query<Movie>().Any(m => m.Id == Id) == false)
            {
                return Request.CreateResponse(HttpStatusCode.NotFound, "Record Not found with the ID supplied");
            }

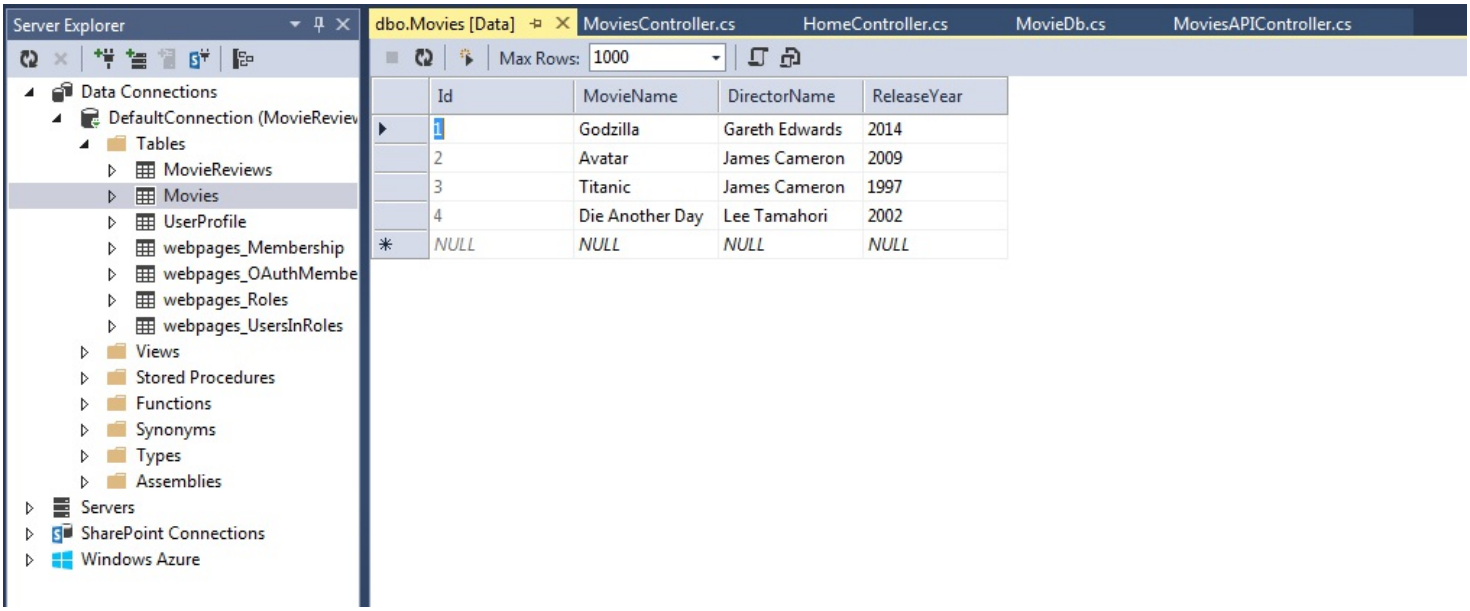
            //else delete
            if (_movieDb.Query<Movie>().Any(m => m.Id == Id))
            {
                Movie movie = context.Movies.Find(Id);
                context.Movies.Remove(movie);
                context.SaveChanges();
            }
        }
    }
    catch (Exception ex)
    {
```

Then, it just returned the 200 status code, saying record deleted and all ok.

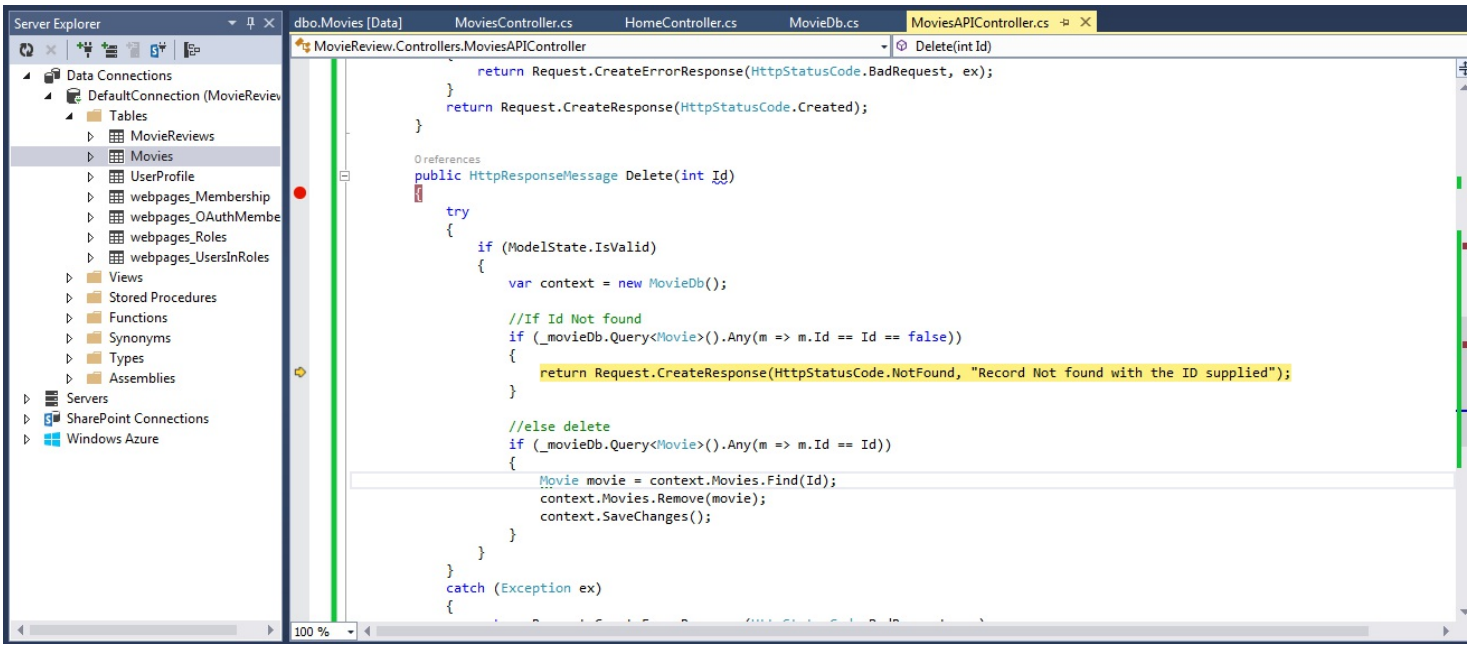


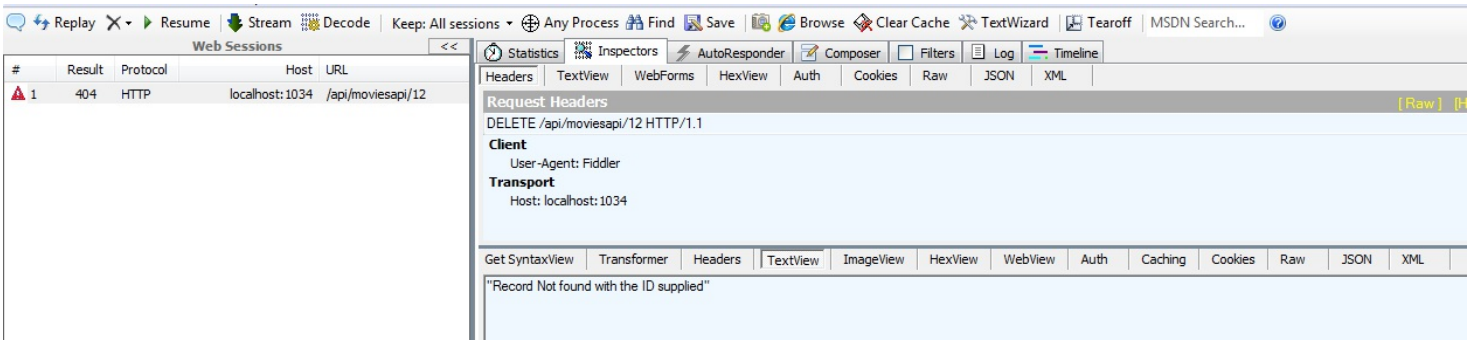
Now, you can verify the same in the

browser as well or in the database as shown below.



Now, since I have already deleted the record and when the same record supplied again, then it will produce the below result.





Now, let's go ahead and check the PUT Implementation for the same.

Implementing PUT:-

So, basically you are going to use PUT,

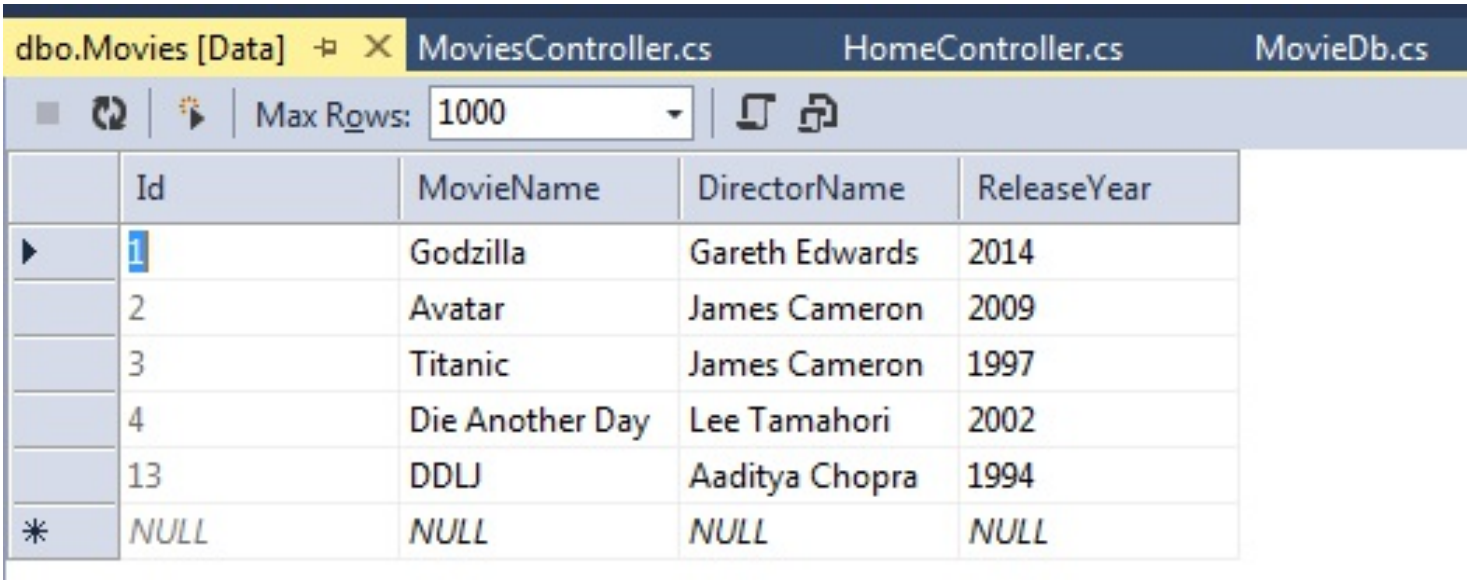
when you need to update some resource. This is also simple implementation. So, as you can see in the below snippet, it is 1st checking whether the id supplied is valid or not.

```
public HttpResponseMessage Put(int Id, [FromBody] Movie movie)
        {
            try
            {
                if (ModelState.IsValid)
                {
                    var context = new MovieDb();

                    //If Id Not found
                    if (_movieDb.Query<Movie>().Any(m => m.Id == Id) == false)
                    {
                        Return Request.CreateResponse(HttpStatusCode.NotFound,
"Record Not found with the ID supplied");
                    }

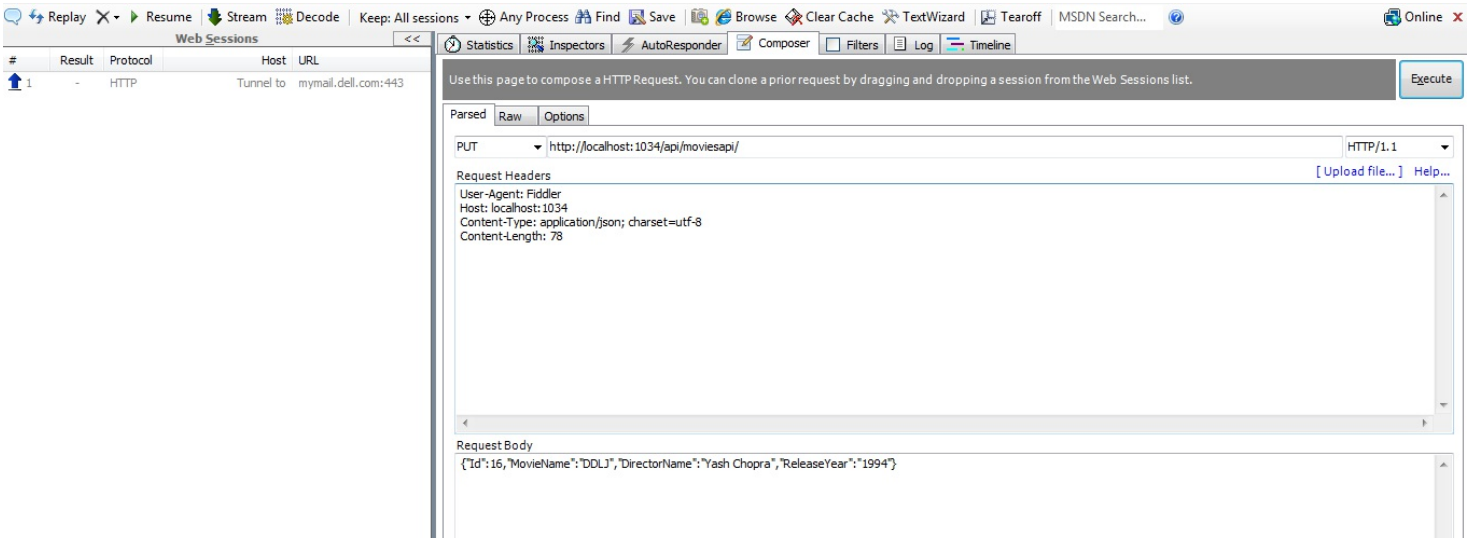
                    //If Id found, then go ahead and update the record
                     if (_movieDb.Query<Movie>().Any(m => m.Id == Id))
                        {
                            context.Movies.Add(movie);
                            context.SaveChanges();
                        }
                    }
                 }
            catch (Exception ex)
            {
                return Request.CreateErrorResponse(HttpStatusCode.BadRequest, ex);
            }
            return Request.CreateResponse(HttpStatusCode.OK);
        }
```

Now, as you can see in the below screenshot, I have updated the record with id = 13, as shown below.

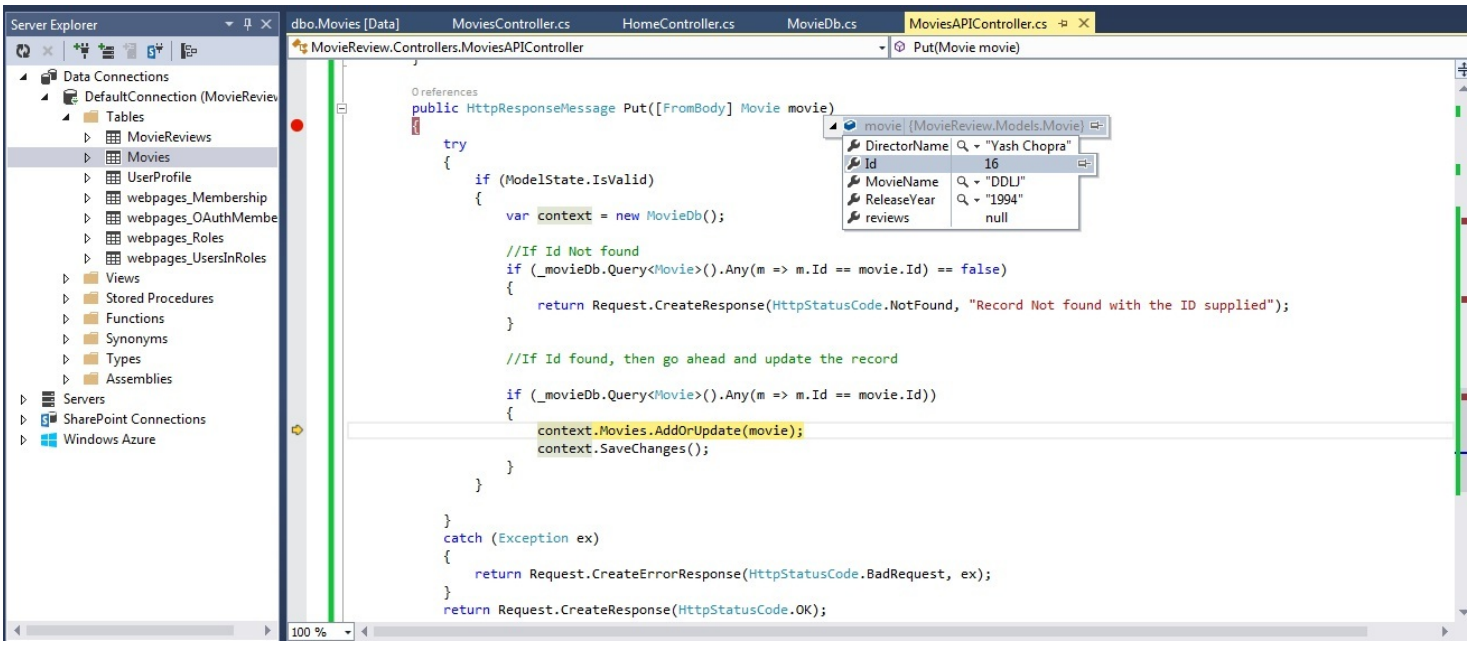


Id	MovieName	DirectorName	ReleaseYear
1	Godzilla	Gareth Edwards	2014
2	Avatar	James Cameron	2009
3	Titanic	James Cameron	1997
4	Die Another Day	Lee Tamahori	2002
13	DDLJ	Aaditya Chopra	1994
NULL	*NULL*	*NULL*	*NULL*

So, here I'll go ahead and change the director name to "***Yash Chopra***" from fiddler as shown below.

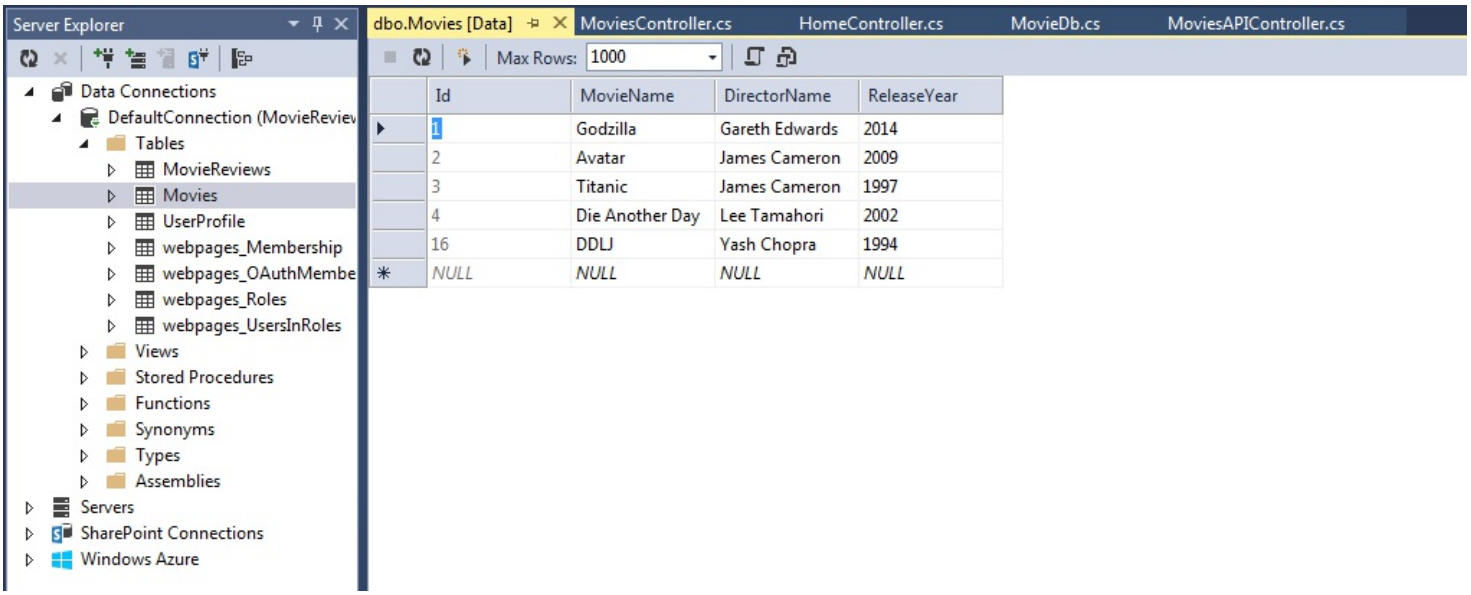


Now, when I execute, then it will come in the debugger and update the record as shown below.



Then, it will just update the record as

shown below.



So, here we have seen Web APIs in action. Now, with this I would just like to wrap this session here.

Summary:-

So, in this chapter, we have seen some of the quite useful features supported by MVC like Web API and Dependency

Injection. We have implemented Web API right from the scratch in our project. We have also seen what are the pre-requisites required to Implement Web API. We have also implemented Dependency Injection to simplify the process. We have used third party component like Ninject to implement the same. Now, in the coming chapter, we'll take this base model and make the site in such a way that it should support Mobile sites as well.

Chapter 13: Making a Mobile Site Using JQuery Mobile

WHAT DO you find in this CHAPTER?

⇒ Introduction

⇒ Setup

⇒ Viewports, Widgets and Attributes

⇒ Themes and Swatches

⇒ Media Query

⇒ Layouts and Pages

⇒ Collapsibles

⇒ Grids and Buttons

⇒ Forms

⇒ Summary

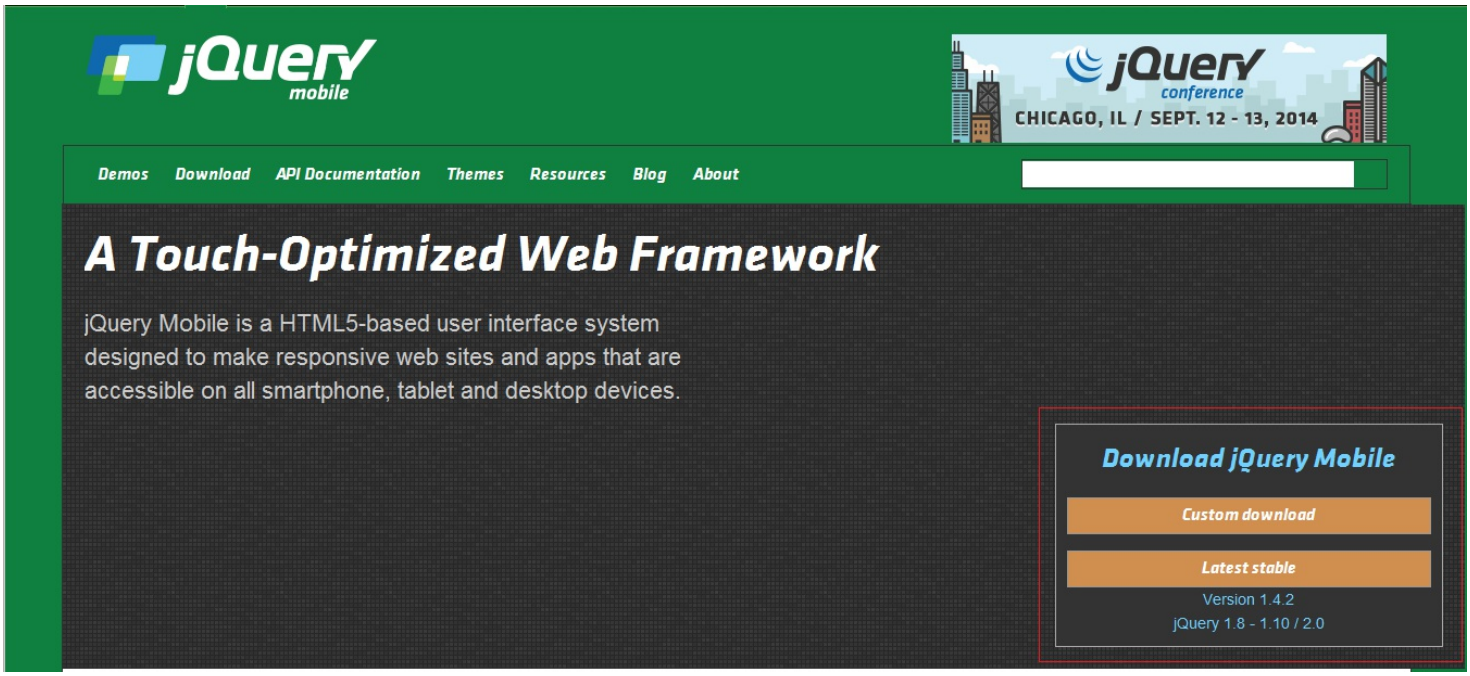
Introduction:-

Hi Friends, now in this section we are going to cover JQuery Mobile feature. JQuery mobile is a JavaScript library which builds on the top of popular JQuery library. So, it allows you to build web applications which work on all type of **Mobiles**, **Desktops**, **Laptops** and many other devices. So, in this section, we are going to add mobile feature in the existing ASP.Net MVC application which we have built so far. So, the library required to support these features is available in "***JqueryMobile***" which can be downloaded from **http://Jquerymobile.com**. It is also known as "***JQM***". Now, JQM provides

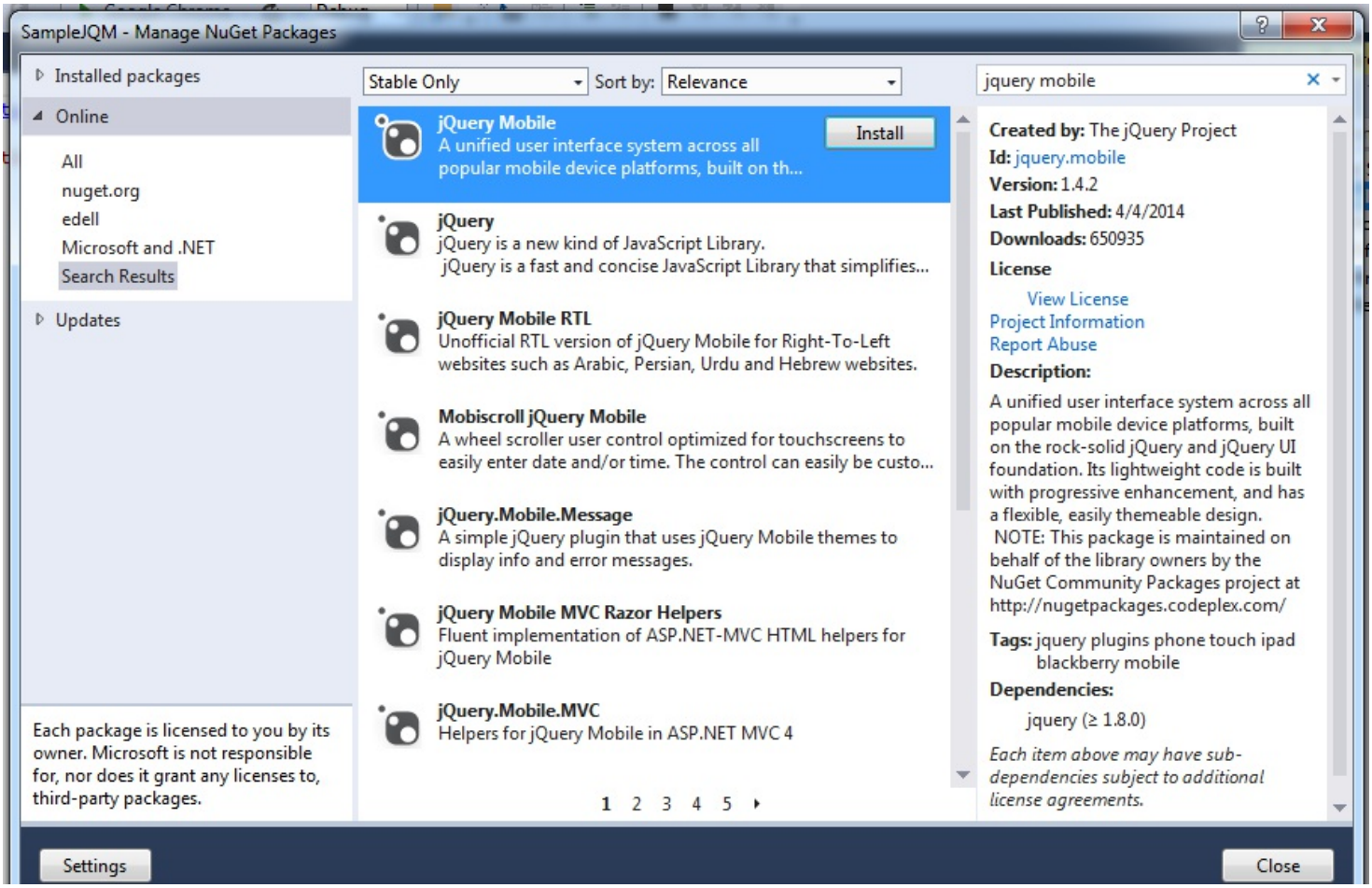
you a unified user interface system which means it does have some set of rules which you have to obey throughout. Most of the hard work is taken care by JQM itself. Features like Async Page Navigation, Progressive Enhancement, Responsive Design, Touch Support, Theming etc. are supported by default. So, let's go ahead and get started.

Setup:-

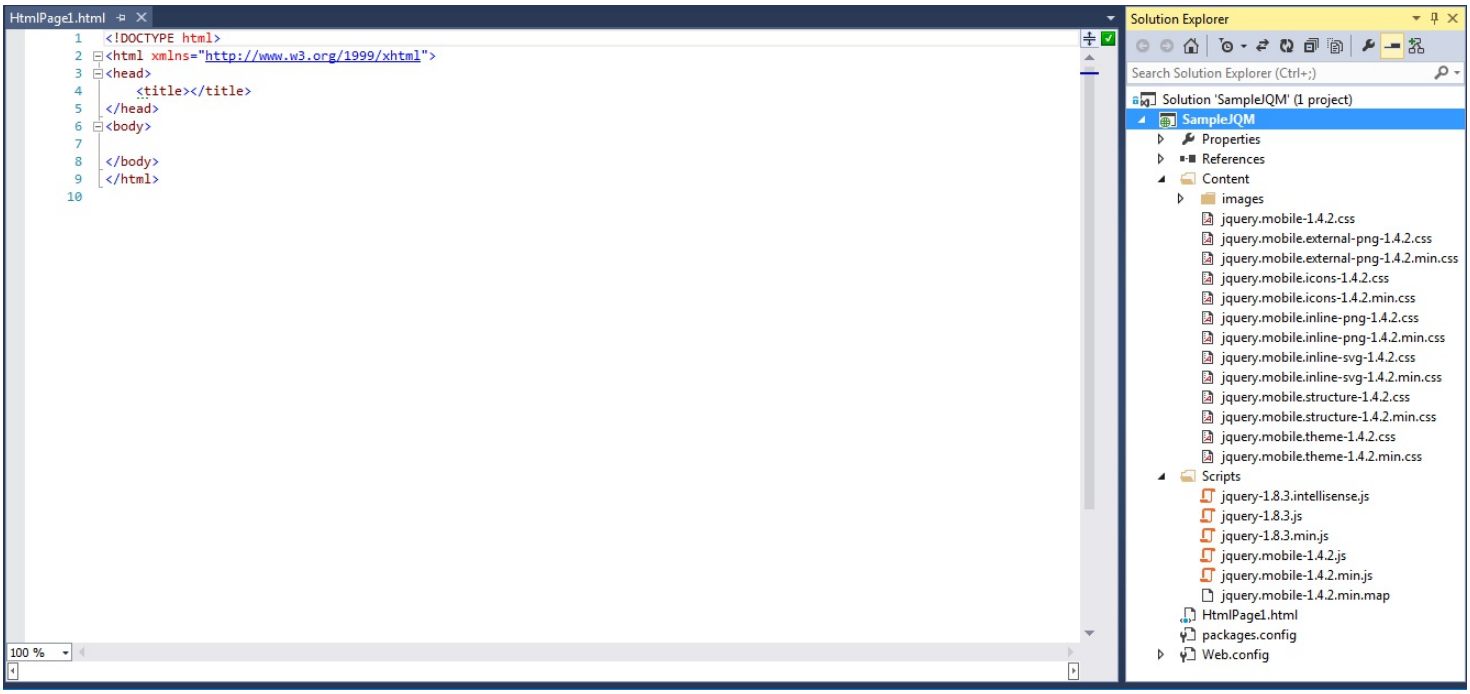
So, obviously before jumping into development of mobile features; we need to download the library 1st. so, one way to download the scripts directly from the JQM site as shown below.



Now, before directly jumping to convert the existing application into JQM version, let's understand few basics with a sample plain example. So, here as you can see below, I have one plain html page in a fairly simple web application. Now, obviously my favorite way of installing any third party software via nuget itself.



So, once you have successfully installed the package, you could see the related contents in your solution explorer.

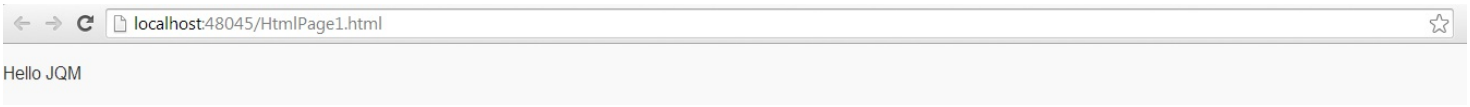


So, in order to get started, we need to include style sheets and scripts on the page as shown below.



Now, if I simply go ahead and run the app, then it will produce the below

output.

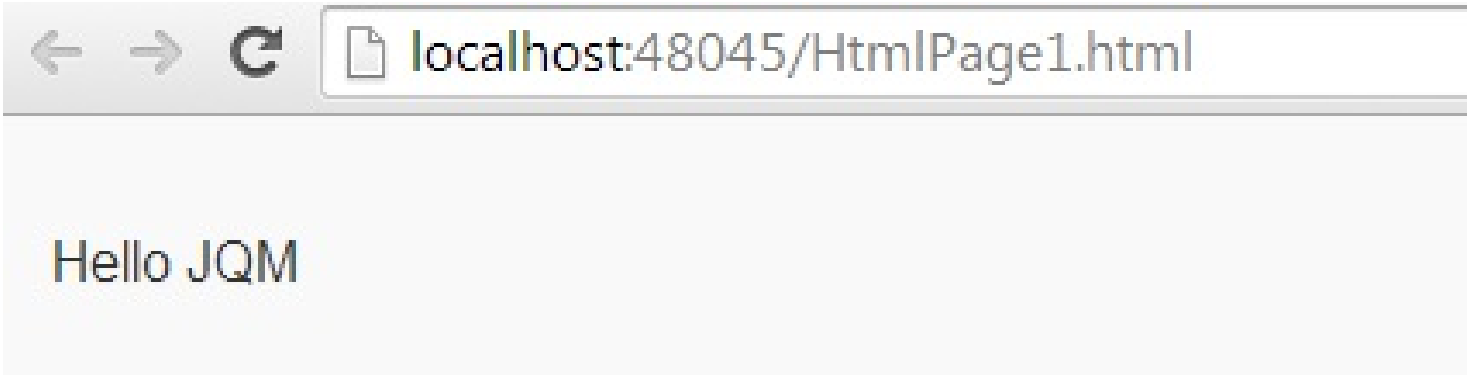


So, as you could see in the screen shot, JQM theme applied to that "<*p*>" element. But, in order to harness the actual potential of JQM; let's apply more settings. Now, let me modify the code a bit, now I am going to provide certain widget as seen below.



Now, these **data-role** applied

page/content wise. Now, if I go ahead and refresh the page again, you will see visual difference applied to the content.



So, these **Data –(dash)** attributes are actually HTML 5 specifications. So, this says that you can apply any number of custom specifications as you want. Also, these data- attributes work well with older browser as well. So, in order to get more idea about HTML 5 specifications, you can refer "***http://www.w3.org/TR/html5***". So, here our scripts are capturing these data

– attributes and outputting the results. So, obviously you can add your custom attributes and write the script to handle the same. Now, let me add one header here, to at least get the look and feel off Mobile menus.

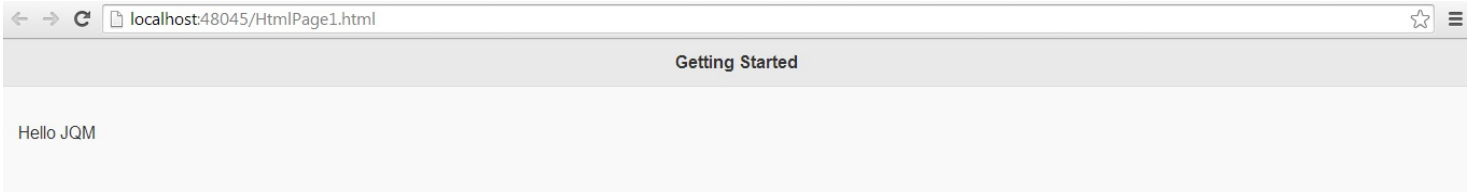
```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>

        <link href="Content/jquery.mobile-1.4.2.css" rel="stylesheet" />
        <script src="Scripts/jquery-1.8.3.js"></script>
        <script src="Scripts/jquery.mobile-1.4.2.js"></script>
        <title>Hello JQM</title>
    </head>
<body>

    <div data-role="page">
        <div data-role="header">
            <h1>Getting Started</h1>
        </div>
        <div data-role="content">
            <p>Hello JQM</p>
        </div>
    </div>
</body>
</html>

```

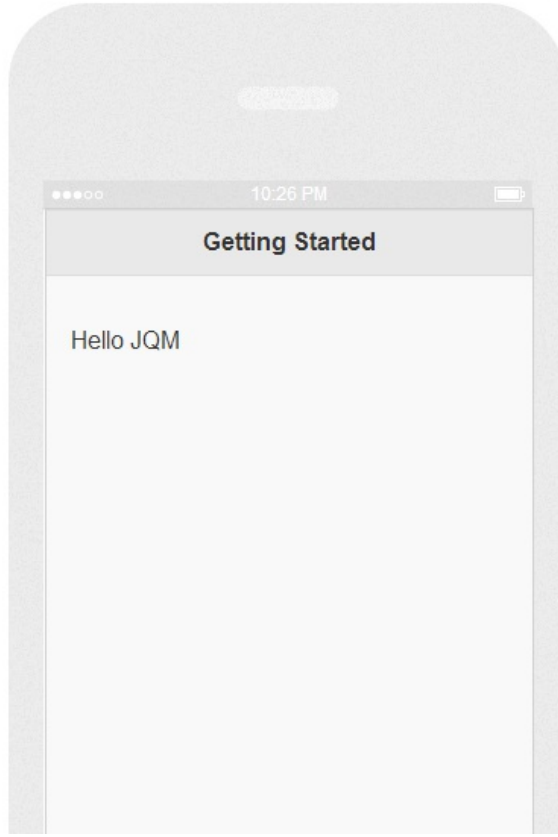
Now, when I save the same and refresh the page, it will produce the below output.

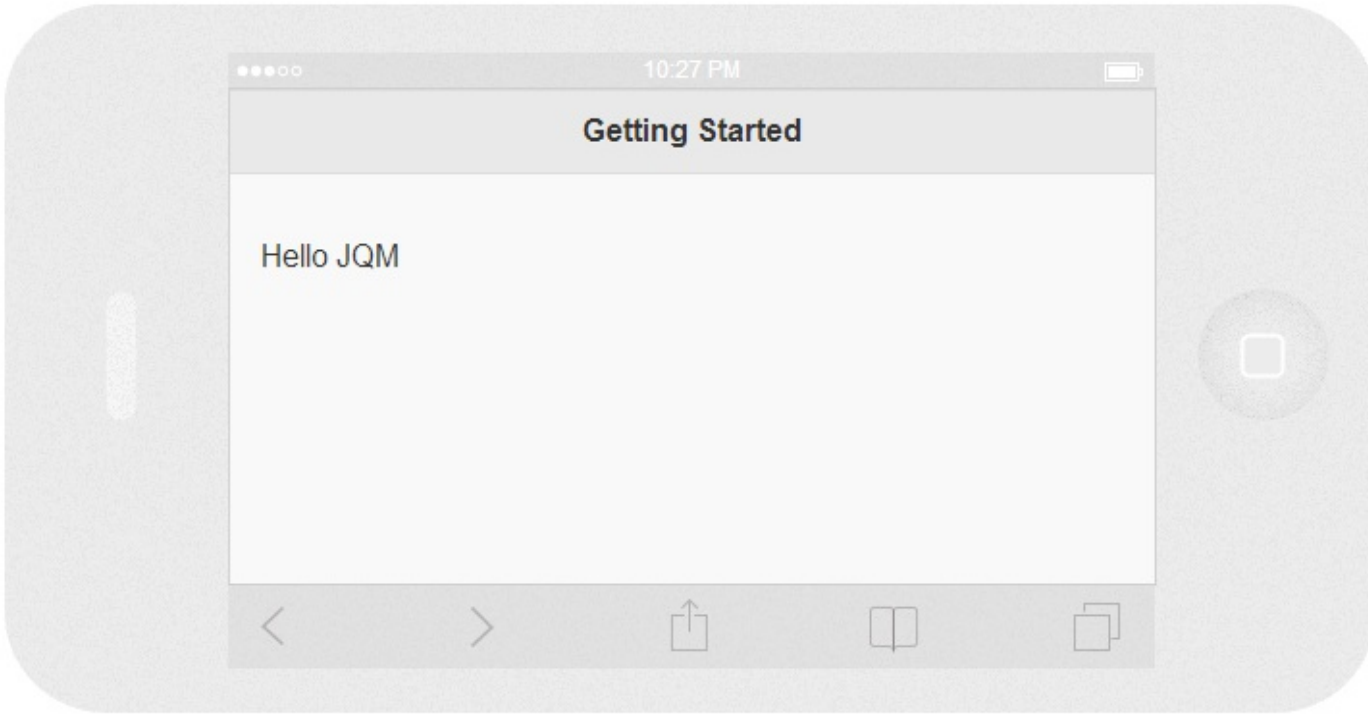


View Ports, Widgets and Attributes:-

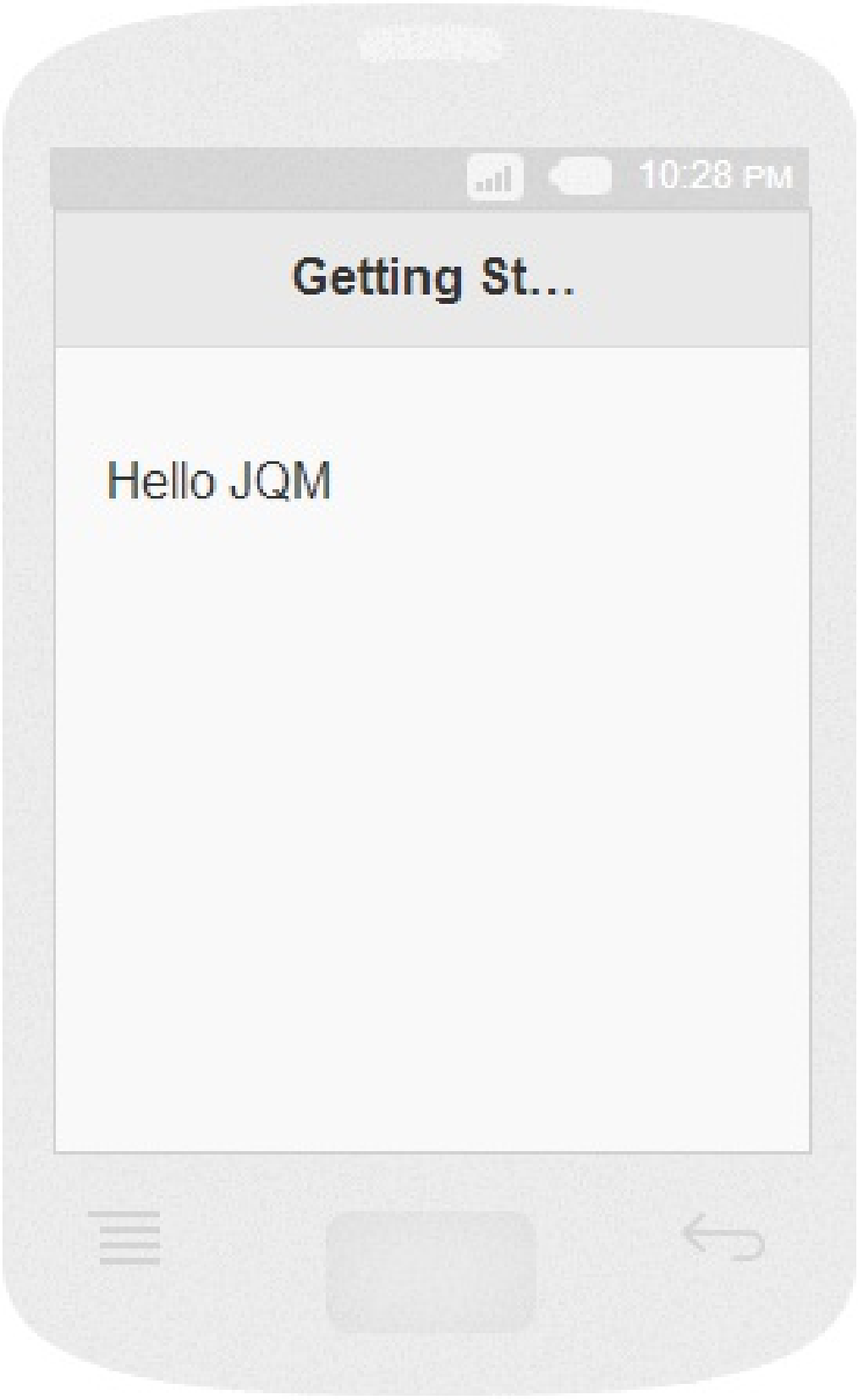
Now, let me go ahead an open the url in (***http://www.responsinator.com***). Now, **Responsinator** is basically a testing site to test your UX in different browsers with different virtual devices. So, as you can see below in the screen shot, I have pasted my localhost URL in there and ran it, then it loaded different views in different devices, and apparently it appears to me, that the content is zoomed out, is not pretty to eyes.

www.responsinator.com/?url=http%3A%2F%2Flocalhost%3A48045%2FHtmlPage1.html





iPhone 3+4 landscape · width: 480px



Crappy Android portrait · width: 240px

Now, by default all the web pages are designed basically to be best viewed in the desktop sized screens, hence when you are seeing the same in smaller devices, it appears to be very small in size. Now, in order to fix this we have a very fine solution and that is called "***ViewPort***". Now, view port Meta tag is basically an instruction for web browsers, so they are going to detect this setting and do the same as specified in this Meta tag. Also the tag allows you to control the size and scale of the viewport. So, in the below view port definition I made sure that Zoom level is set to 1, which means neither zoomed in nor zoomed out and also set the width to device width which means it will get

automatically adjusted whatever device it gets opened.

HtmlPage1.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta name="viewport" content="width=device-wdith,initial-scale=1"/>
        <link href="Content/jquery.mobile-1.4.2.css" rel="stylesheet" />
        <script src="Scripts/jquery-1.8.3.js"></script>
        <script src="Scripts/jquery.mobile-1.4.2.js"></script>
        <title>Hello JQM</title>
    </head>
<body>

    <div data-role="page">
        <div data-role="header">
            <h1>Getting Started</h1>
        </div>
        <div data-role="content">
            <p>Hello JQM</p>
        </div>
    </div>
</body>
</html>

```

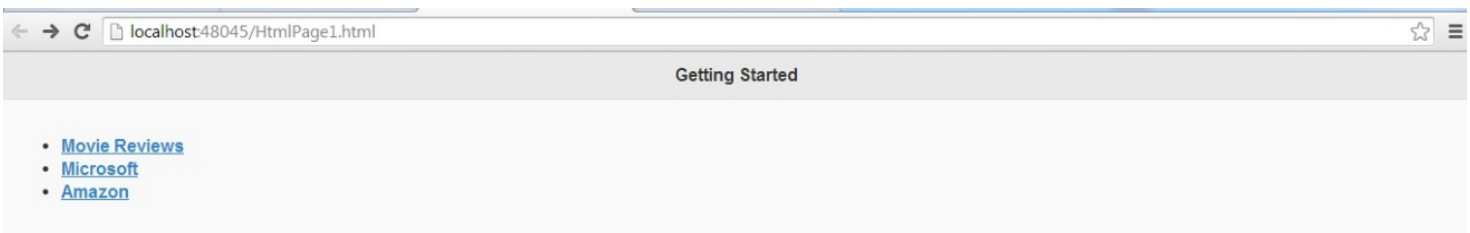
Now, this Meta tag is only detected by the mobile browsers, which are not going to make any difference to desktop version of the page. Now, let me go ahead and add some more content to this page. Let's add menus in there.

HtmlPage1.html

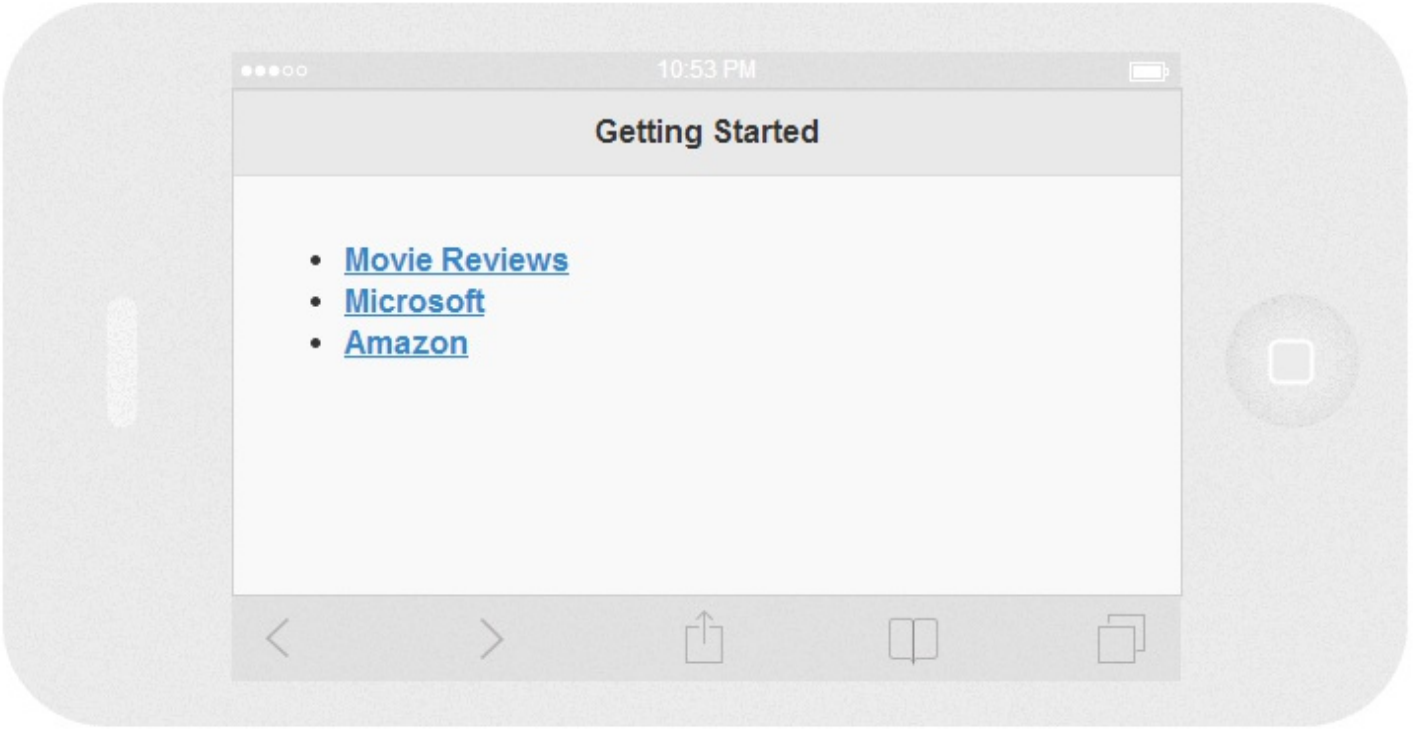
```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta name="viewport" content="width=device-wdith,initial-scale=1"/>
        <link href="Content/jquery.mobile-1.4.2.css" rel="stylesheet" />
        <script src="Scripts/jquery-1.8.3.js"></script>
        <script src="Scripts/jquery.mobile-1.4.2.js"></script>
        <title>Hello JQM</title>
    </head>
<body>

    <div data-role="page">
        <div data-role="header">
            <h1>Getting Started</h1>
        </div>
        <div data-role="content">
            <ul>
                <li><a href="http://mymoviereviews.azurewebsites.net/">Movie Reviews</a></li>
                <li><a href="http://www.microsoft.com">Microsoft</a></li>
                <li><a href="http://www.amazon.com">Amazon</a></li>
            </ul>
        </div>
    </div>
</body>
</html>
```

So, in desktop, it would look like below

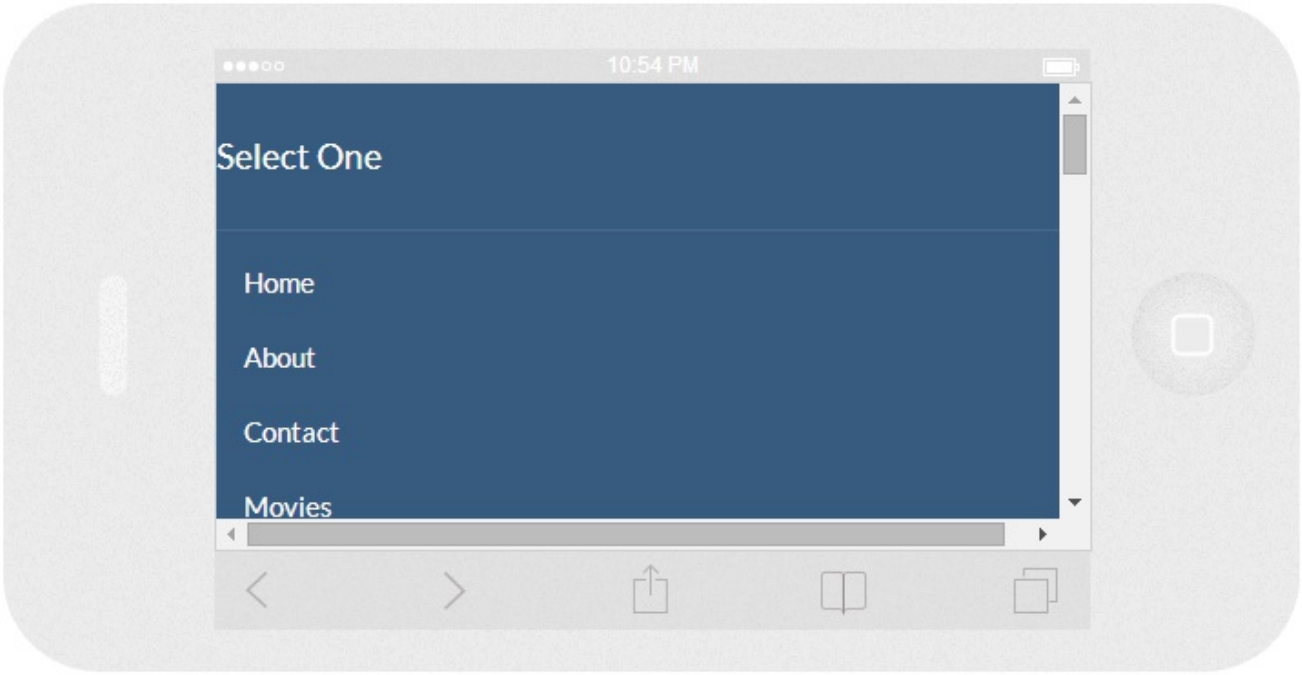


But, in mobile browser, it would appear like



iPhone 3+4 landscape · width: 480px

And responsinator is smart enough to detect your events. So, if you click on any of these links, it will open here as shown below.



iPhone 3+4 landscape · width: 480px

So, currently the site which we have developed is not best optimized for web. But I will do in this segment. But, for now just stick to the basics. Now, currently I can see unordered list in the browser, but I can also assign the same with data role list view as shown below.

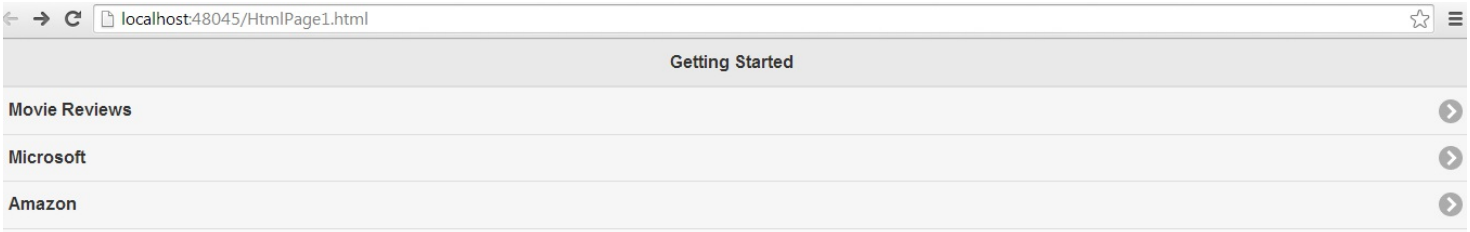
HtmlPage1.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta name="viewport" content="width=device-wdith,initial-scale=1"/>
        <link href="Content/jquery.mobile-1.4.2.css" rel="stylesheet" />
        <script src="Scripts/jquery-1.8.3.js"></script>
        <script src="Scripts/jquery.mobile-1.4.2.js"></script>
        <title>Hello JQM</title>
    </head>
<body>

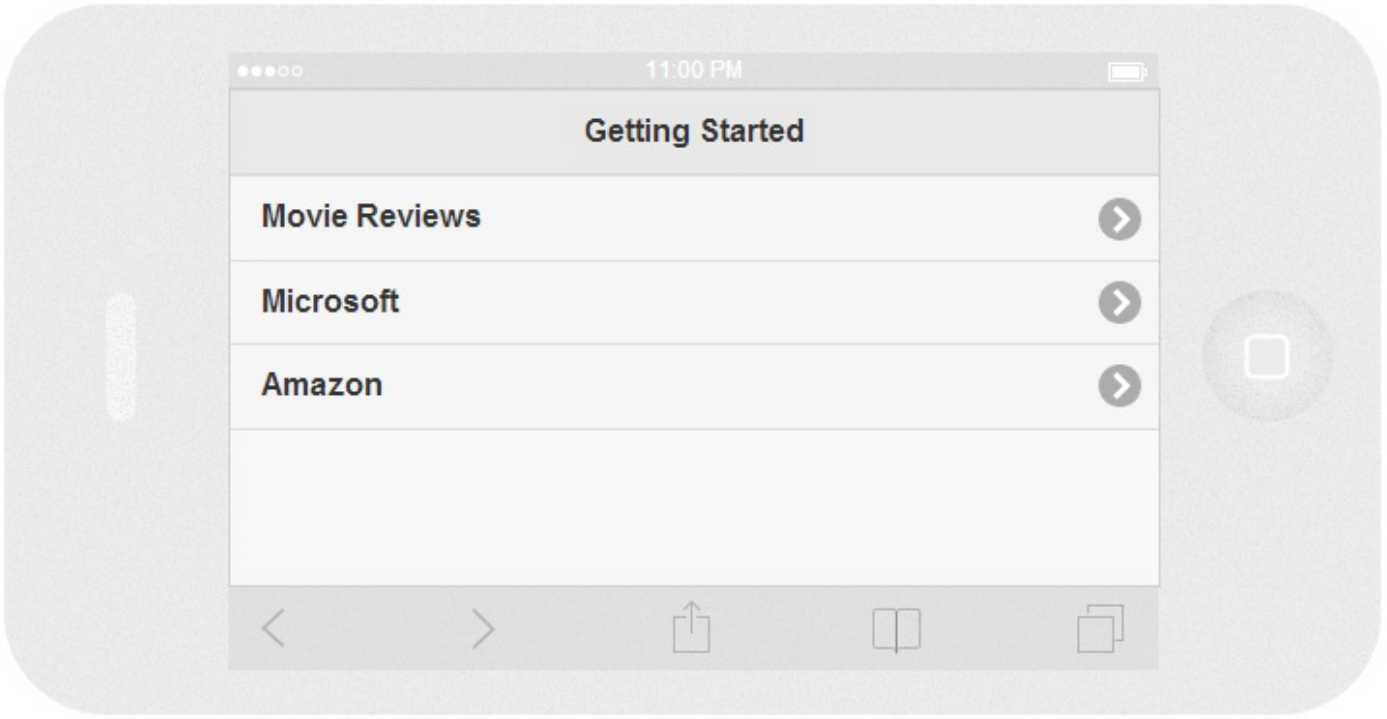
    <div data-role="page">
        <div data-role="header">
            <h1>Getting Started</h1>
        </div>
        <div data-role="content">
            <ul data-role="listview">
                <li><a href="http://mymoviereviews.azurewebsites.net/">Movie Reviews</a></li>
                <li><a href="http://www.microsoft.com">Microsoft</a></li>
                <li><a href="http://www.amazon.com">Amazon</a></li>
            </ul>
        </div>
    </div>
</body>
</html>

```

Now, when I save the page and refresh the same, it will produce the below effect.



Now, if you open the same in mobile browser, it will appear like below



iPhone 3+4 landscape · width: 480px

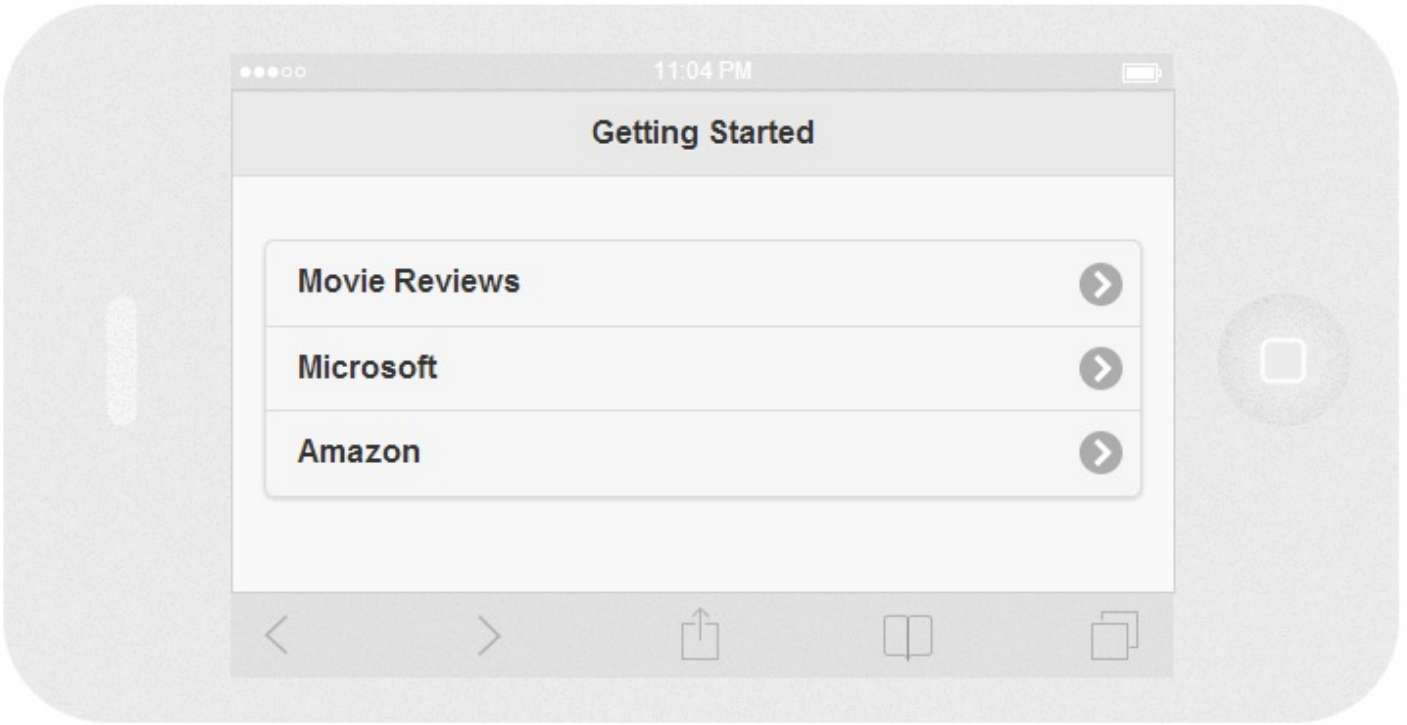
So, here it will also have hover effect when you hover on any of these icons. Now, I can also apply some margins and rounded corners by using widget as shown below.

HtmlPage1.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta name="viewport" content="width=device-wdith,initial-scale=1"/>
        <link href="Content/jquery.mobile-1.4.2.css" rel="stylesheet" />
        <script src="Scripts/jquery-1.8.3.js"></script>
        <script src="Scripts/jquery.mobile-1.4.2.js"></script>
        <title>Hello JQM</title>
    </head>
<body>

    <div data-role="page">
        <div data-role="header">
            <h1>Getting Started</h1>
        </div>
        <div data-role="content">
            <ul data-role="listview" data-inset="true">
                <li><a href="http://mymoviereviews.azurewebsites.net/">Movie Reviews</a></li>
                <li><a href="http://www.microsoft.com">Microsoft</a></li>
                <li><a href="http://www.amazon.com">Amazon</a></li>
            </ul>
        </div>
    </div>
</body>
</html>
```

Now, when I save the same refresh the browser, I can see the effect.



iPhone 3+4 landscape · width: 480px

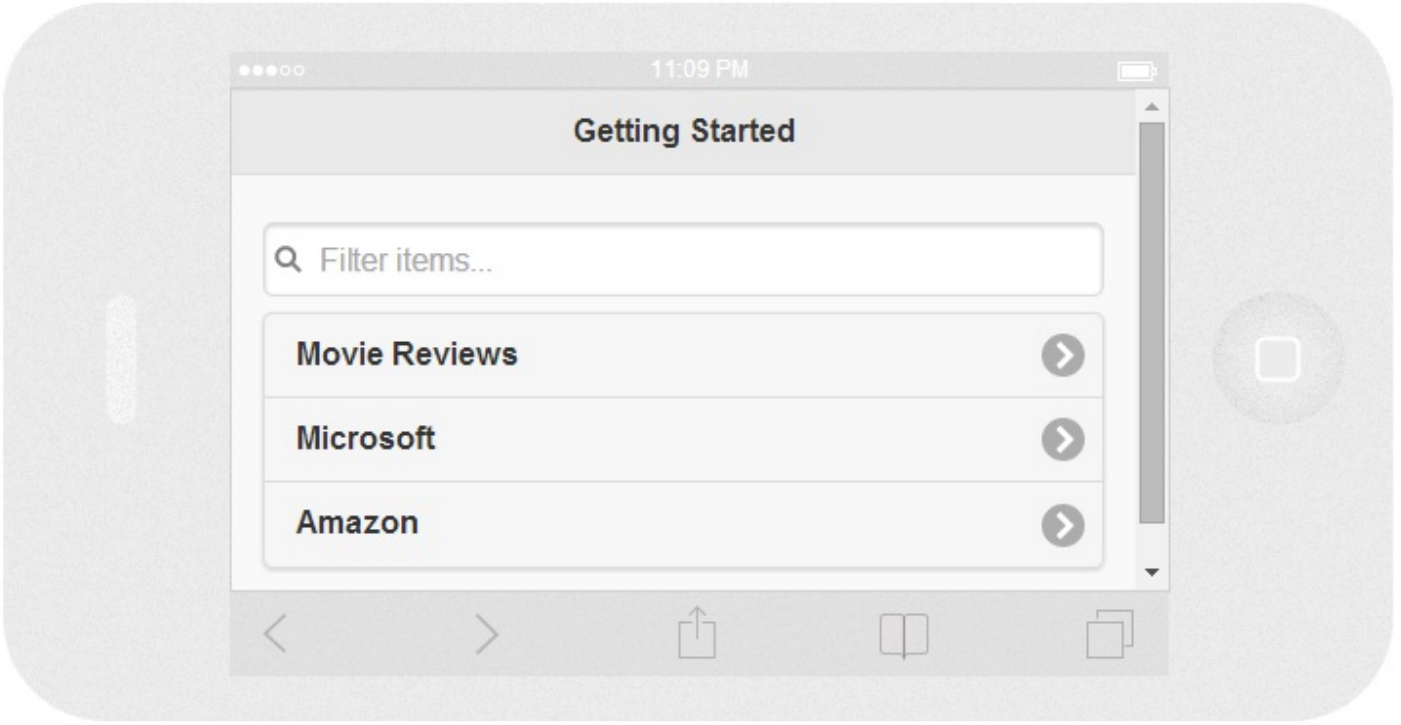
Here, I can also go ahead and add some filtering option by using the "***data*** –" attribute as shown below.

HtmlPage1.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta name="viewport" content="width=device-wdith,initial-scale=1"/>
        <link href="Content/jquery.mobile-1.4.2.css" rel="stylesheet" />
        <script src="Scripts/jquery-1.8.3.js"></script>
        <script src="Scripts/jquery.mobile-1.4.2.js"></script>
        <title>Hello JQM</title>
    </head>
<body>

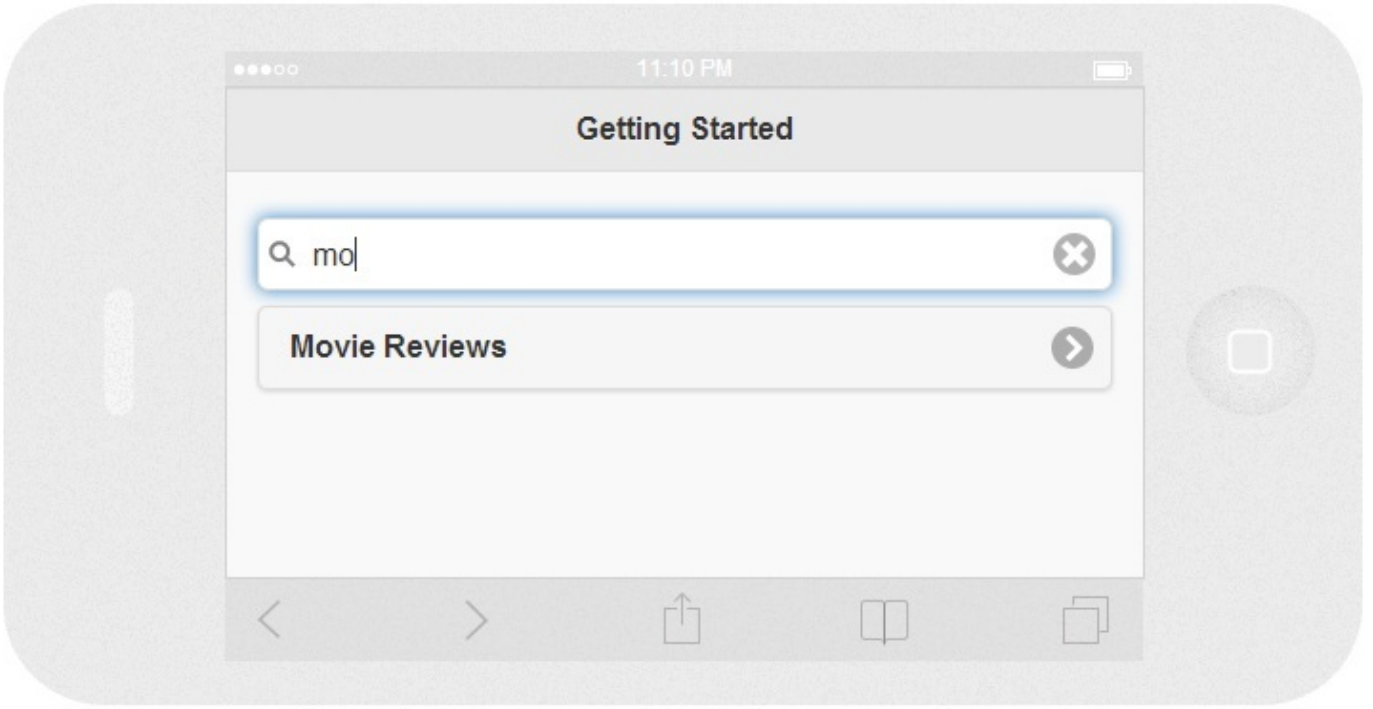
    <div data-role="page">
        <div data-role="header">
            <h1>Getting Started</h1>
        </div>
        <div data-role="content">
            <ul data-role="listview" data-inset="true" data-filter="true">
                <li><a href="http://mymoviereviews.azurewebsites.net/">Movie Reviews</a></li>
                <li><a href="http://www.microsoft.com">Microsoft</a></li>
                <li><a href="http://www.amazon.com">Amazon</a></li>
            </ul>
        </div>
    </div>
</body>
</html>
```

So, now this will add filtering option with watermark in it.



iPhone 3+4 landscape · width: 480px

And, here you don't have to write any script to filter the content; this is already handled in JQM. So, if you type in some values, then it will work like shown below

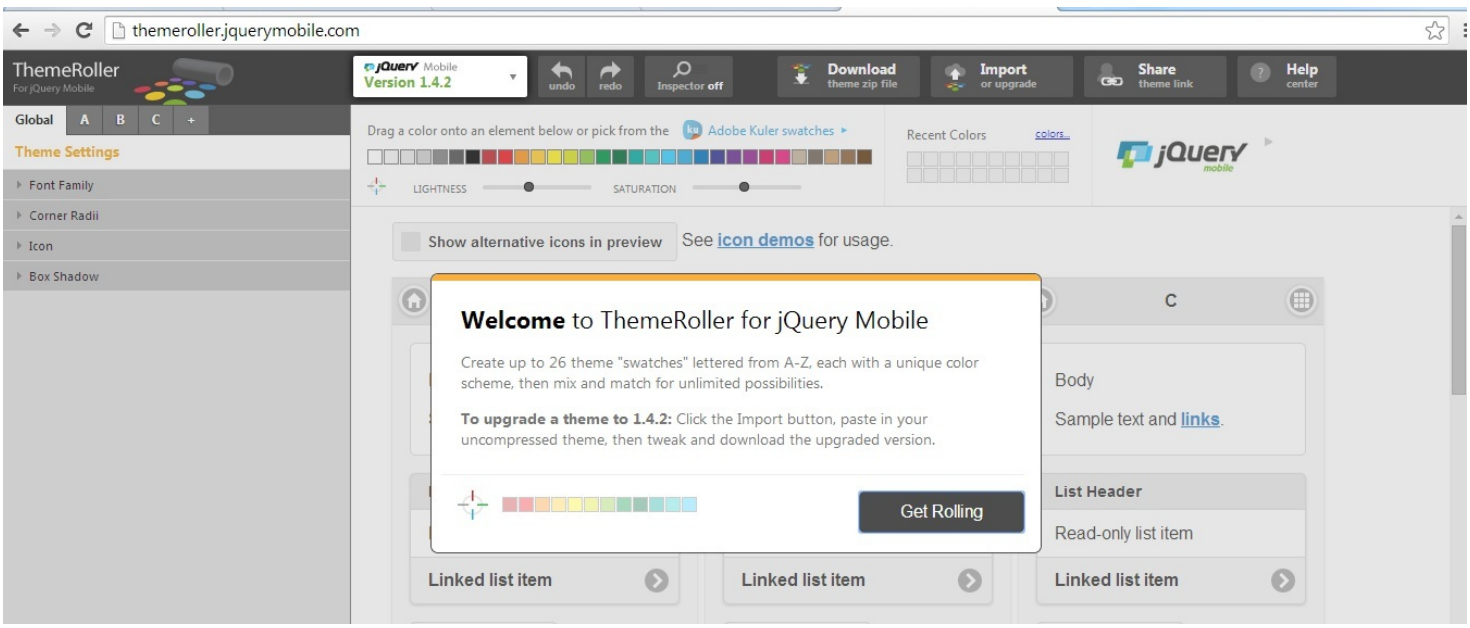


iPhone 3+4 landscape · width: 480px

Themes and Swatches:-

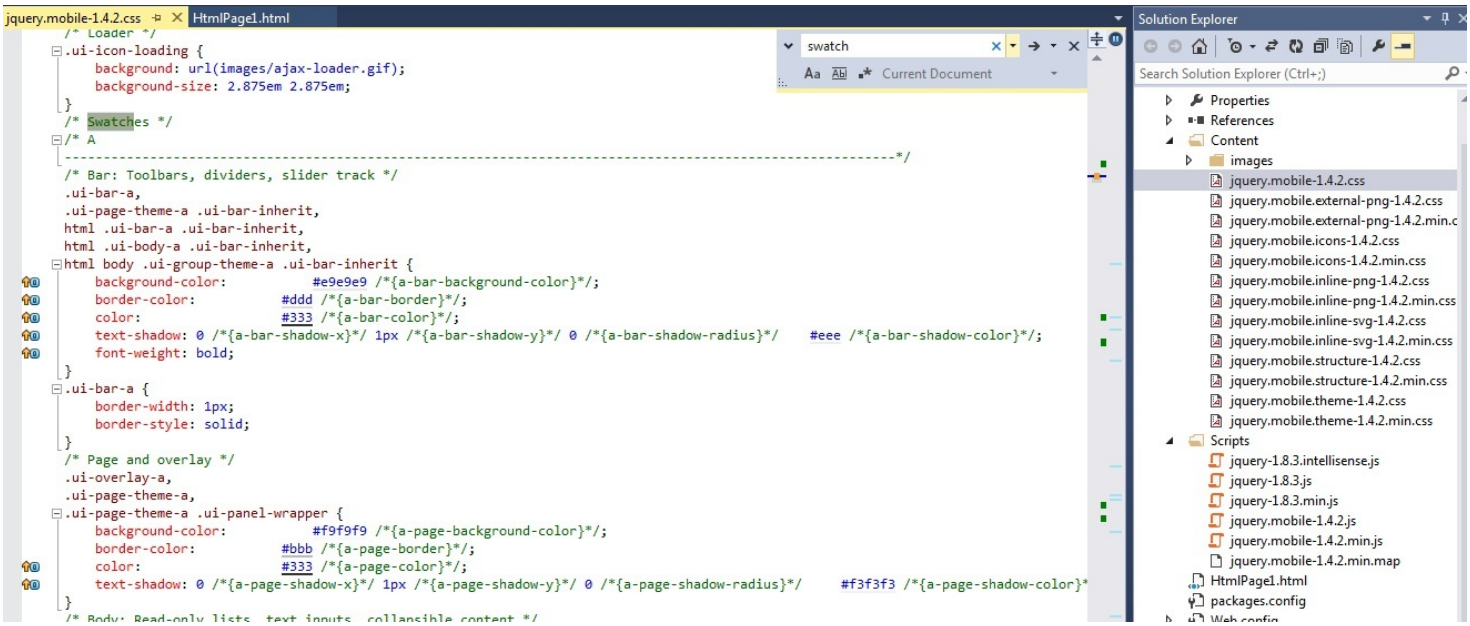
So, at any point if you don't like the theme and you want to change the same, certainly you can do that. So, basically JQuery Framework comes with a theme framework which is implemented in CSS and it supports 26 different color

combinations.



So, as you can see in the above screen shot, you can pick any of these themes and apply the same in your site. Now, these sets of color are called "***Swatches***" and has letters associated with them like "***A, B, C, …***". Now, here you can customize your color combinations with color picker tool as well. So, in your application if you open

the CSS file which we have included in our web page and search with the text "***Swatch***", you will come to below shown point



So, like these we have many settings for different swatches. Now, swatch is nothing but a set of CSS rules for different visual appearances. So by default our application is using "***Swatch A***" theme. But, now I would like to go

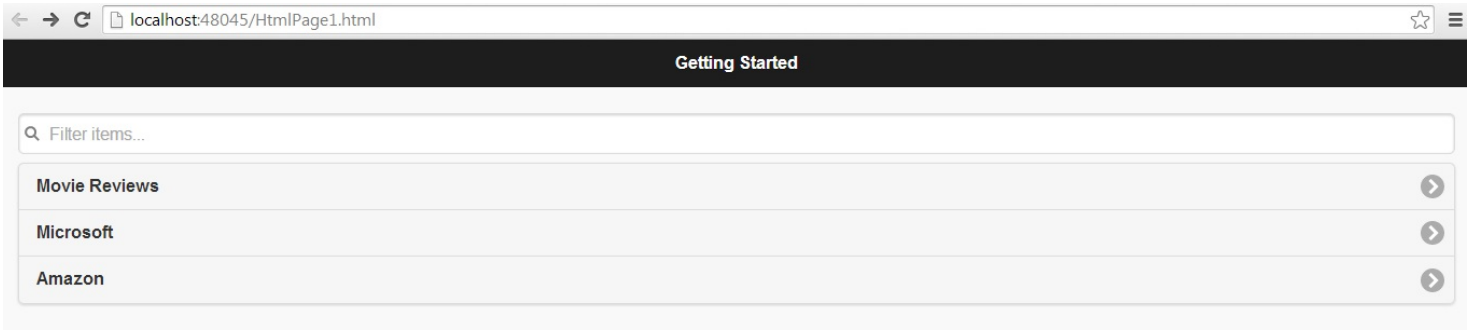
ahead and override one of the swatches. Let me override it for header. Now, in header I would like to use black theme. So, for doing this I will again use data-attribute as shown below.

HtmlPage1.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta name="viewport" content="width=device-wdith,initial-scale=1"/>
        <link href="Content/jquery.mobile-1.4.2.css" rel="stylesheet" />
        <script src="Scripts/jquery-1.8.3.js"></script>
        <script src="Scripts/jquery.mobile-1.4.2.js"></script>
        <title>Hello JQM</title>
    </head>
<body>

    <div data-role="page">
        <div data-role="header" data-theme="b">
            <h1>Getting Started</h1>
        </div>
        <div data-role="content">
            <ul data-role="listview" data-inset="true" data-filter="true">
                <li><a href="http://mymoviereviews.azurewebsites.net/">Movie Reviews</a></li>
                <li><a href="http://www.microsoft.com">Microsoft</a></li>
                <li><a href="http://www.amazon.com">Amazon</a></li>
            </ul>
        </div>
    </div>
</body>
</html>
```

Now, when you save the same and refresh the page, it will produce you the below output, quite pleasing to eyes.



Media Query:-

Now, in this module what I am going to do is use all the above discussed concepts and many more things in the website which we have developed so far. We are also going to optimize the mobile experience in our existing site which means I open the same site on a desktop that will behave like a desktop one or when I open the same site on mobile that will behave like a mobile site. Now, here to fix my site I'll go in

the layout page and apply this Meta tag as shown below.

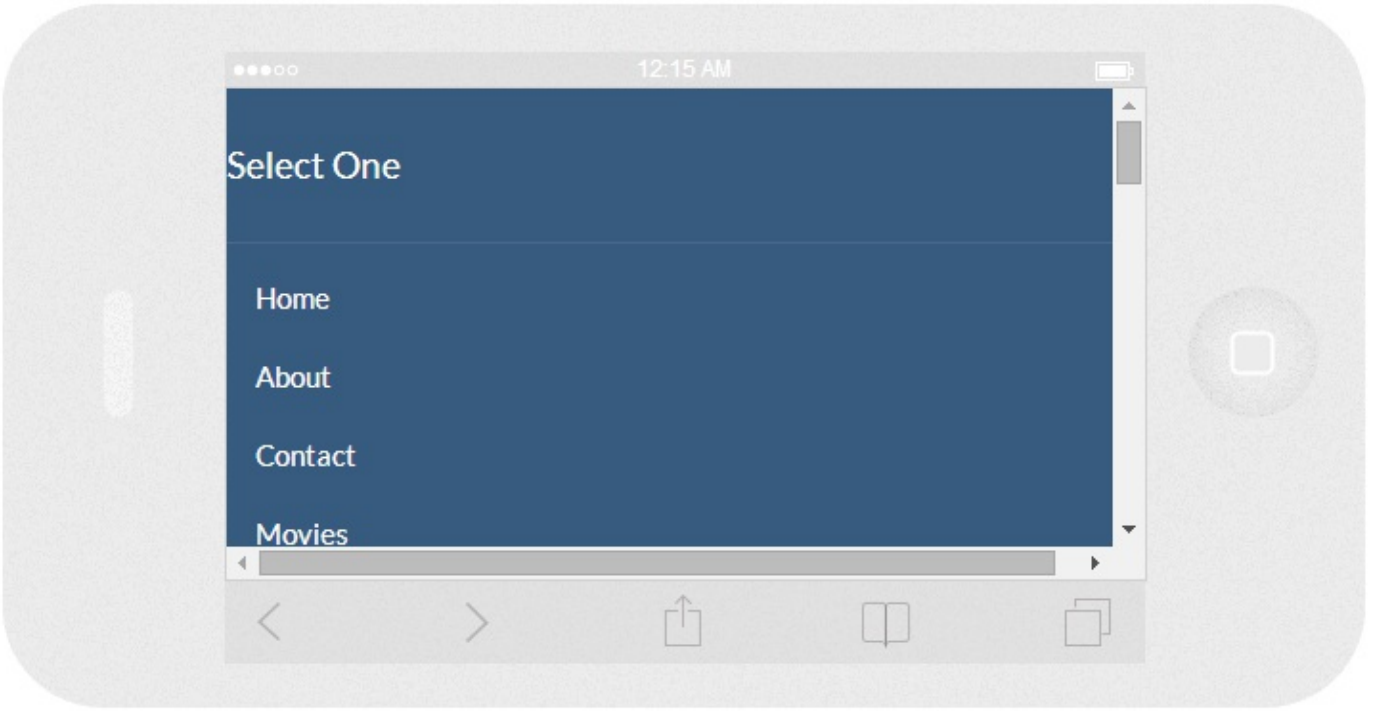
_Layout.cshtml

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>@ViewBag.Title - My ASP.NET MVC Application</title>
    <link href="~/favicon.ico" rel="shortcut icon" type="image/x-icon" />
    <meta name="viewport" content="width=device-width,initial-scale=1.0" />
    <link href="~/Content/bootstrap.min.css" rel="stylesheet" />
    <script src="~/Scripts/jquery-2.1.0.min.js"></script>
    <script src="~/Scripts/bootstrap.js"></script>
    <link href="~/Content/BootStrap-theme-movie.css" rel="stylesheet" />
    @*@Styles.Render("~/Content/css")*@
    @Scripts.Render("~/bundles/modernizr")
</head>

<div class="container">
    <div class="row">
        <div class="navbar navbar-default">
            <div class="navbar-header">
                <ul class="nav navbar-nav">
                    <li><span class="navbar-brand">Select One</span></li>

                </ul>
            </div>
            <div class="navbar-collapse navbar">
                <ul class="nav navbar-nav">
                    <li>@Html.ActionLink("Home", "Index", "Home")</li>
                    <li>@Html.ActionLink("About", "About", "Home")</li>
                    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
                    <li>@Html.ActionLink("Movies", "Index", "Movies")</li>
                    <li>@Html.Partial("_LoginPartial")</li>
                </ul>
```

So, when I save the same and check in the responsinator page, then it will look like as shown below.



iPhone 3+4 landscape · width: 480px

Little bit improvement I can see, but not 100%. So, when I have to write single style sheet which will support both my general desktop and mobile devices, then we have to make certain adjustments. So, in order to achieve the same I'll use the feature "***media query***". Now, a media query is the expression which we can put into a CSS file to

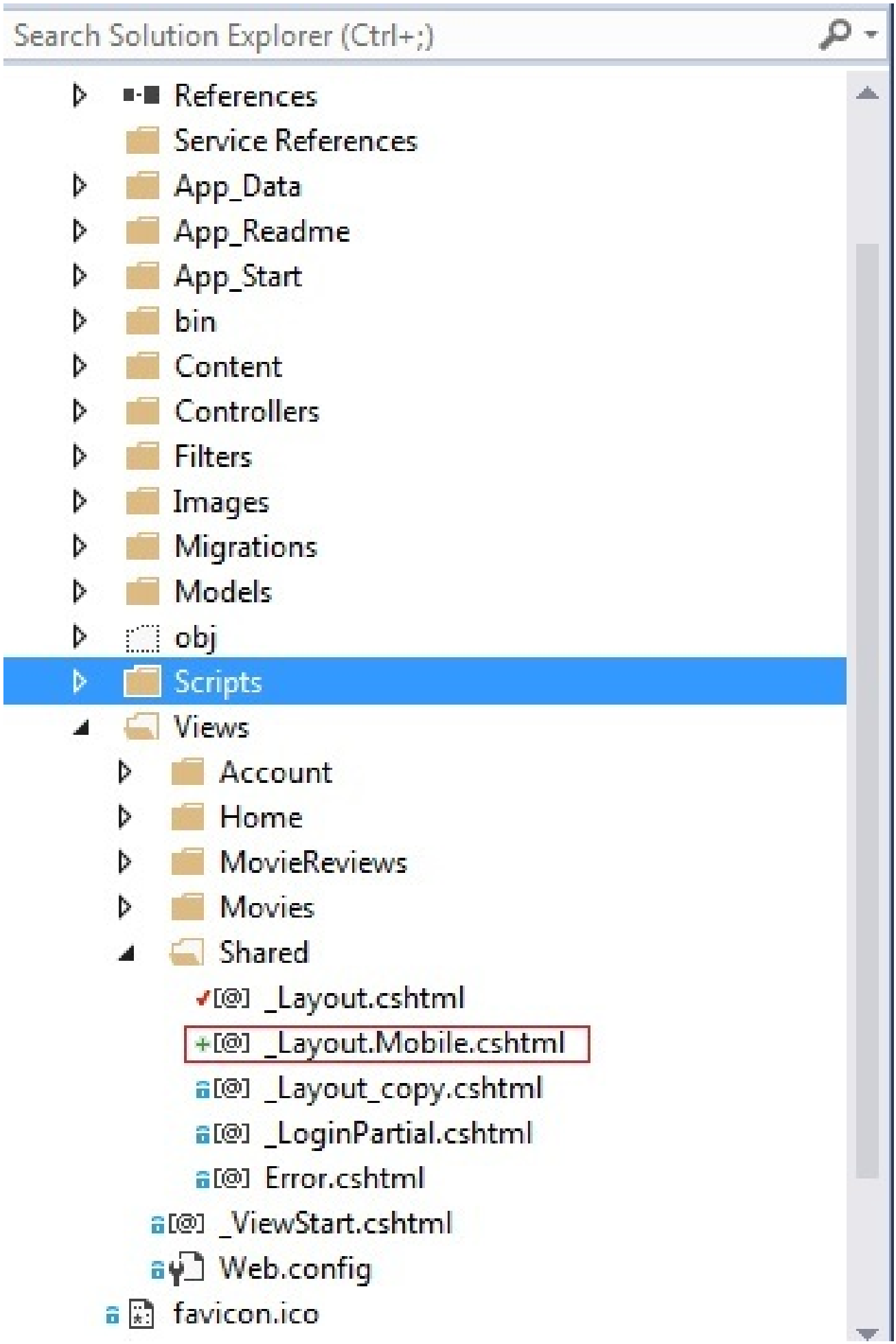
assign style rules based on the media query result. So, basically a media query is simply a query to see what media features are available. So, if the query returns true then style applies otherwise won't apply. But, here in my page I don't have any media which I don't want to render in case of mobile browsers. But, I have just put this concept here to explain one more technique to optimize the look. Below is the sample media query

```
@media screen(min-width: 450px)
{
        #nav li:{
        float: right;
        }
}

@media screen(min-width: 850px)
{
        #nav {
                width: 200px;
        }
        #nav li:{
                float: left;
        }
}
```

Layouts and Pages:-

One of the most common features of MVC 4 is ability to switch views depending on the string which the client sends to the server. Now, the way it works in MVC 4; if the browser is a mobile browser then runtime will render a view ending with "***.mobile***" extension. So, this view switching technique works with all types of views. So, as you can see in the below screen shot I have created one mobile version of layout pages and just added the mobile style sheet and mobile script.

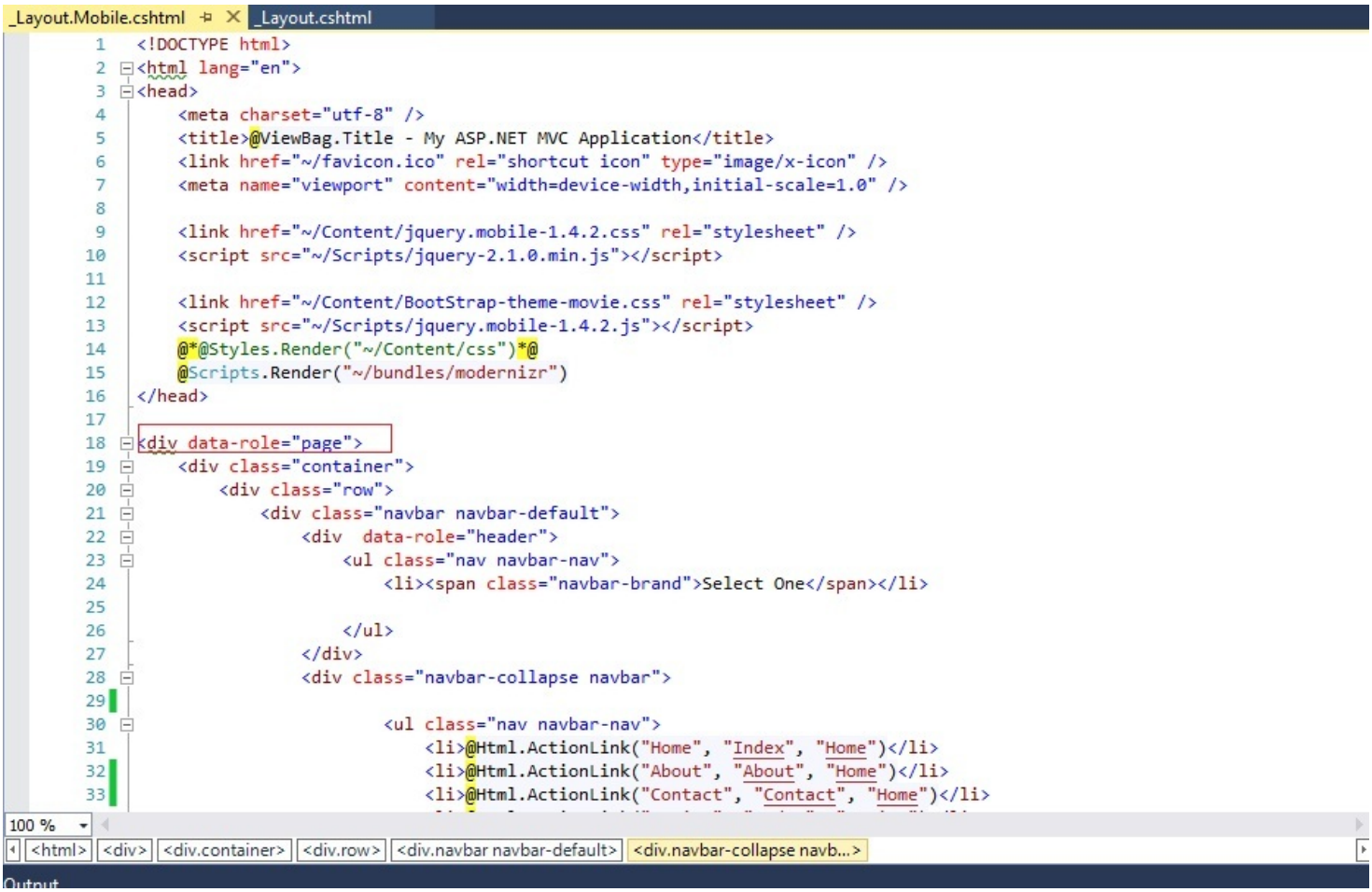

Search Solution Explorer (Ctrl+;)
References
Service References
App_Data
App_Readme
App_Start
bin
Content
Controllers
Filters
Images
Migrations
Models
obj
Scripts
Views
Account
Home
MovieReviews
Movies
Shared
_Layout.cshtml
_Layout.Mobile.cshtml
_Layout_copy.cshtml
_LoginPartial.cshtml
Error.cshtml
_ViewStart.cshtml
Web.config
favicon.ico

_Layout.Mobile.cshtml | _Layout.cshtml

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="utf-8" />
5      <title>@ViewBag.Title - My ASP.NET MVC Application</title>
6      <link href="~/favicon.ico" rel="shortcut icon" type="image/x-icon" />
7      <meta name="viewport" content="width=device-width,initial-scale=1.0" />
8
9      <link href="~/Content/jquery.mobile-1.4.2.css" rel="stylesheet" />
10     <script src="~/Scripts/jquery-2.1.0.min.js"></script>
11
12     <link href="~/Content/BootStrap-theme-movie.css" rel="stylesheet" />
13     <script src="~/Scripts/jquery.mobile-1.4.2.js"></script>
14     @*@Styles.Render("~/Content/css")*@
15     @Scripts.Render("~/bundles/modernizr")
16 </head>
17
18 <div class="container">
19     <div class="row">
20         <div class="navbar navbar-default">
21             <div class="navbar-header">
22                 <ul class="nav navbar-nav">
23                     <li><span class="navbar-brand">Select One</span></li>
24
25                 </ul>
26             </div>
27             <div class="navbar-collapse navbar">
28                 <ul class="nav navbar-nav">
29                     <li>@Html.ActionLink("Home", "Index", "Home")</li>
30                     <li>@Html.ActionLink("About", "About", "Home")</li>
31                     <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
32                     <li>@Html.ActionLink("Movies", "Index", "Movies")</li>
33                     <li>@Html.Partial("_LoginPartial")</li>
```

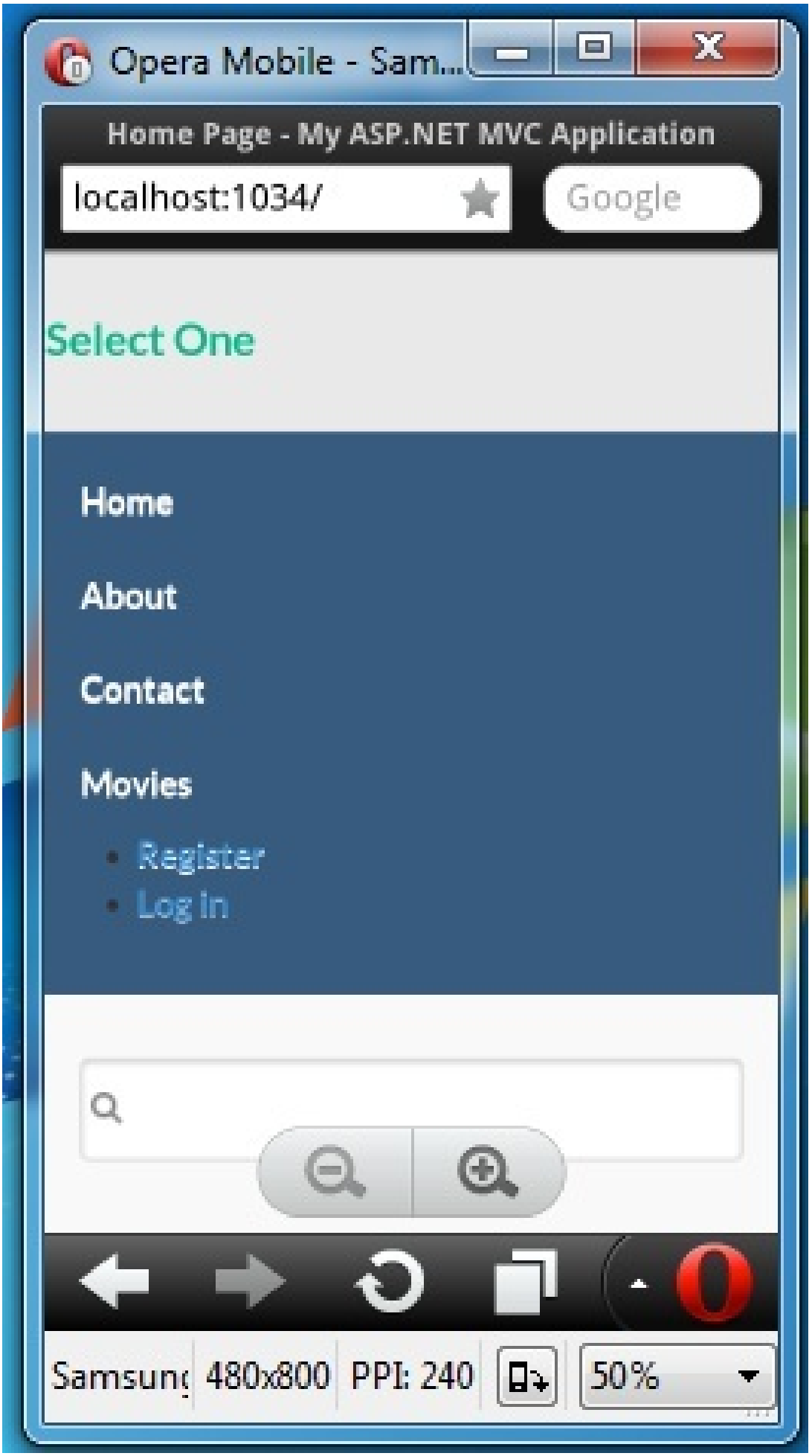
100 %

Now, the next thing which I need to do is adjust some basic settings like adding divs and data-roles to page as shown below.



Now, the closing tag of this div would be at the bottom of the page, just before closing HTML tag. Here one more thing I would like to tell you that, from here I am going to test my mobile features in **Opera Mobile** browser for desktop. So, this is basically an emulator where in you can set various mobile devices

settings and test your site in it. So, I am testing in Samsung-Galaxy setting which you can see below.

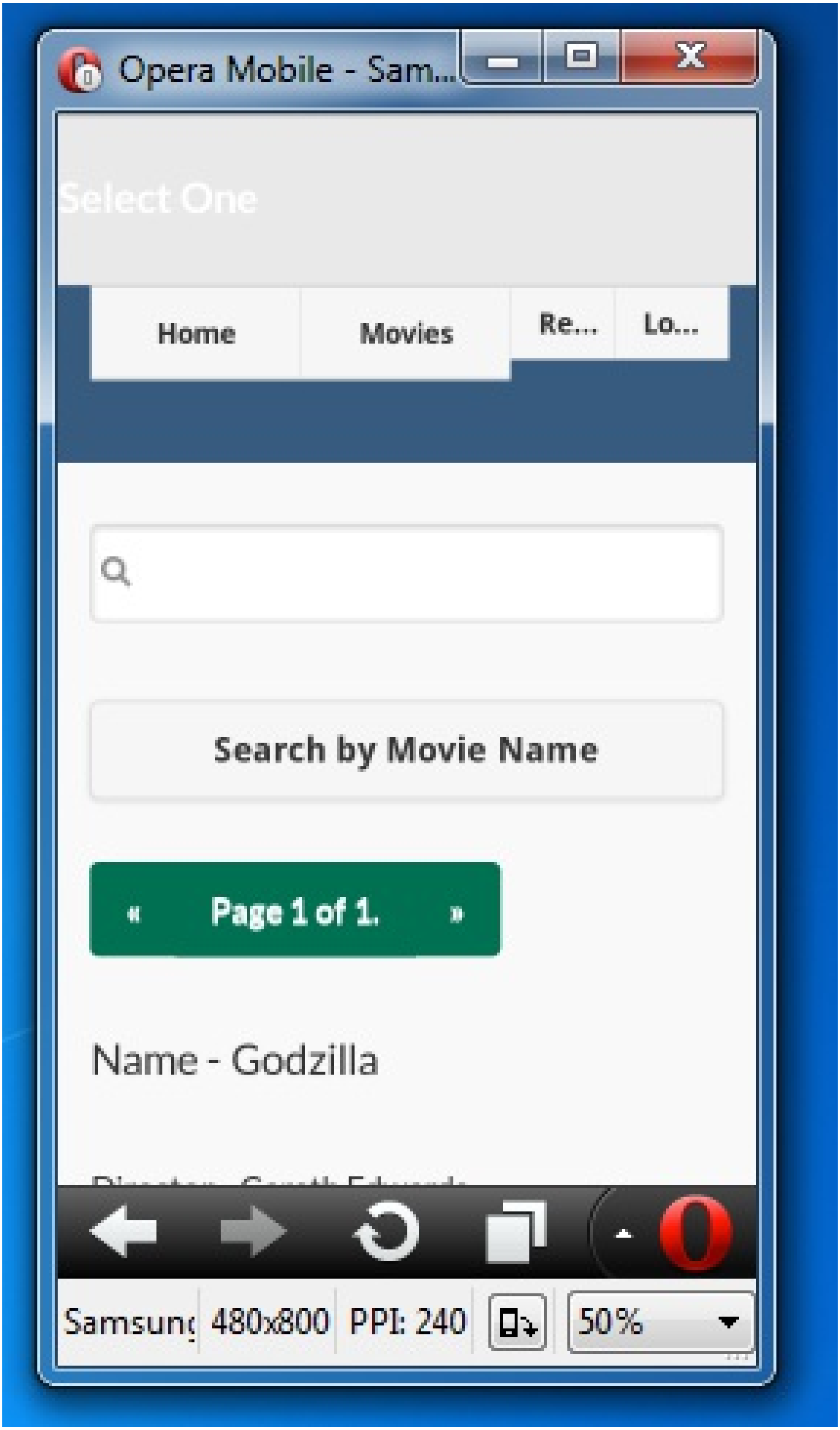

Opera Mobile - Sam...
Home Page - My ASP.NET MVC Application
localhost:1034/
Google
Select One
Home
About
Contact
Movies
Register
Log in
Samsung
480x800
PPI: 240
50%

So, now my menus are vertical, let's do one thing stretch these menus horizontally so that we can have some more space visible on the screen.

```
<div class="navbar-collapse navbar">
    <div data-role="navbar">
        <ul class="nav navbar-nav">
            <li>@Html.ActionLink("Home", "Index", "Home")</li>
            @*<li>@Html.ActionLink("About", "About", "Home")</li>
            <li>@Html.ActionLink("Contact", "Contact", "Home")</li>*@
            <li>@Html.ActionLink("Movies", "Index", "Movies")</li>
            <li>@Html.Partial("_LoginPartial")</li>
        </ul>
    </div>
</div>
```

Now, as you can see in the above screen shot, I have used "***data-role navbar***", to align the menus horizontally. So, this widget is already there with JQM to align the things horizontally. Also, I have also commented unnecessary menus. Now, let's see in opera browser by just

refreshing the same.

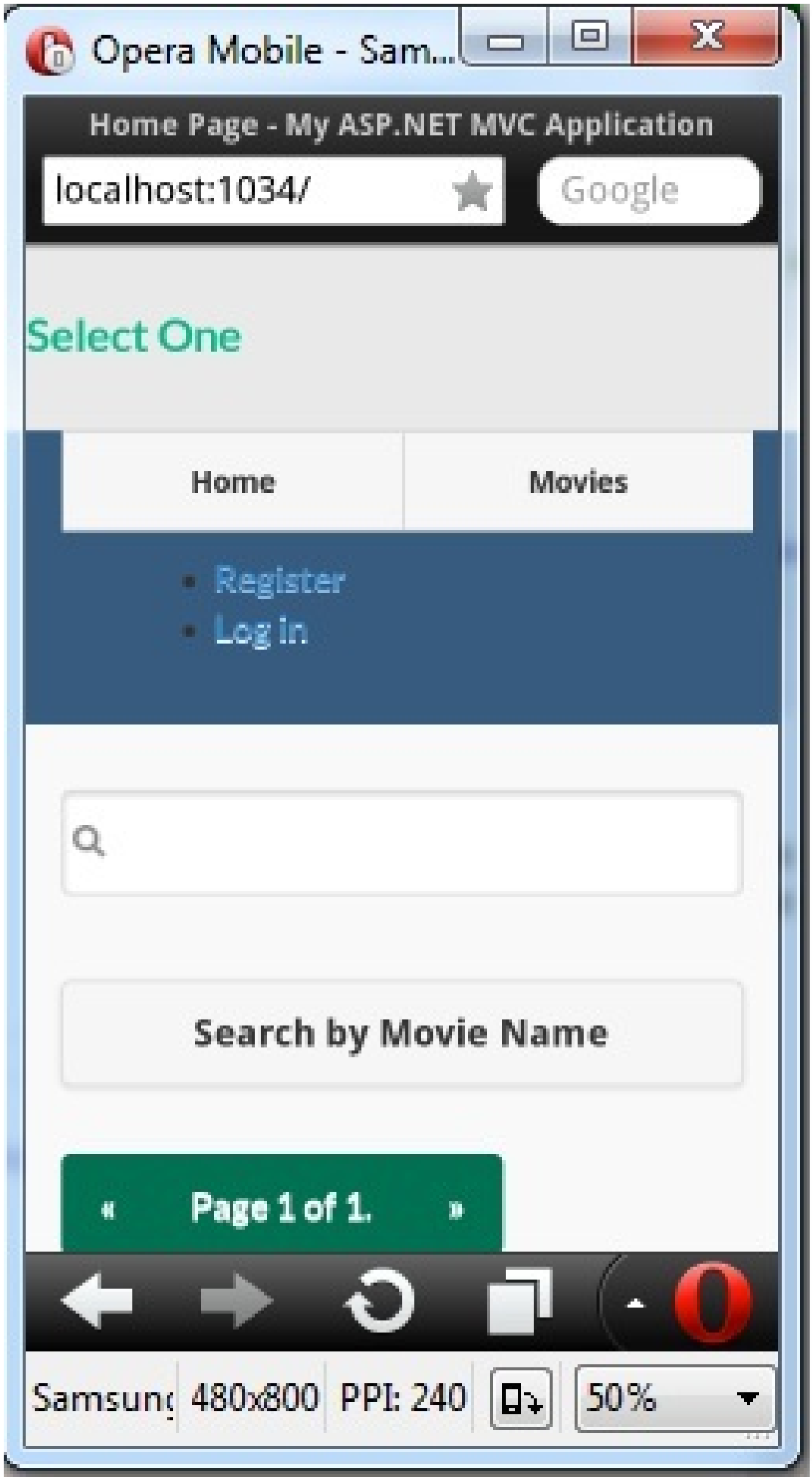

Opera Mobile - Sam...
Select One
Home
Movies
Re...
Lo...
Search by Movie Name
« Page 1 of 1. »
Name - Godzilla
Samsung
480x800
PPI: 240
50%

Now, if you notice this, now, my register and login link not looking perfect. So, what I can do here is I can take the registration and login link out of this navbar and keep the same aligned vertically as shown below.

```
<div class="navbar-collapse navbar">
    <div data-role="navbar">
        <ul class="nav navbar-nav">
            <li>@Html.ActionLink("Home", "Index", "Home")</li>
            @*<li>@Html.ActionLink("About", "About", "Home")</li>
            <li>@Html.ActionLink("Contact", "Contact", "Home")</li>*@
            <li>@Html.ActionLink("Movies", "Index", "Movies")</li>

        </ul>
    </div>
    <ul><li class="nav navbar-nav">@Html.Partial("_LoginPartial")</li></ul>
</div>
```

Then, it will appear quite good as shown below.


Opera Mobile - Sam...
Home Page - My ASP.NET MVC Application
localhost:1034/
Google
Select One
Home
Movies
Register
Log in
Search by Movie Name
« Page 1 of 1. »
Samsung
480x800
PPI: 240
50%

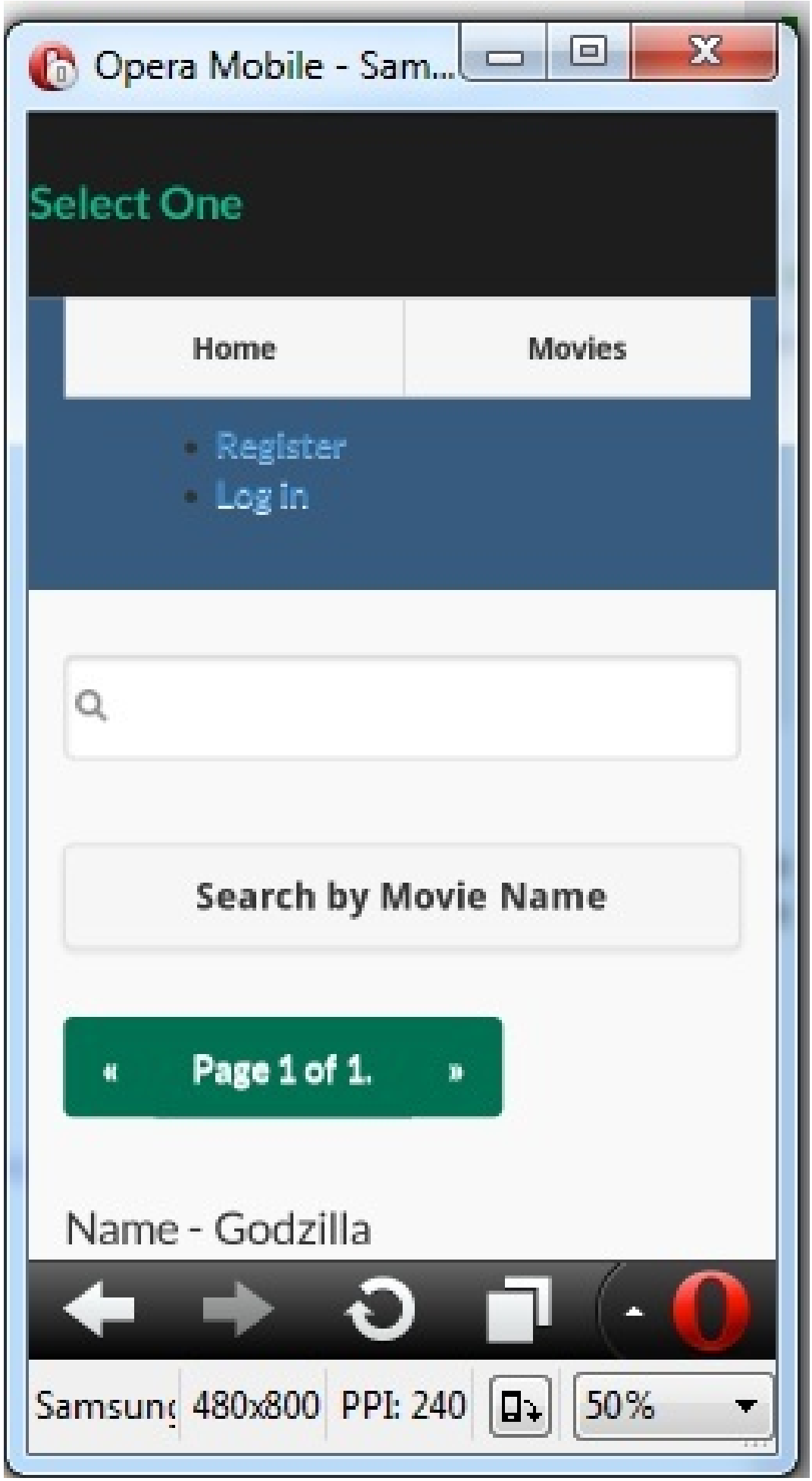
Also, let me change the theme swatch of header to black. So for that I'll just change theme setting as shown below.

```
<div data-role="page">
    <div class="container">
        <div class="row">
            <div class="navbar navbar-default">
                <div  data-role="header" data-theme="b">
                    <ul class="nav navbar-nav">
                        <li><span class="navbar-brand">Select One</span></li>

                    </ul>
                </div>
                <div class="navbar-collapse navbar">
                    <div data-role="navbar">
                        <ul class="nav navbar-nav">
                            <li>@Html.ActionLink("Home", "Index", "Home")</li>
                            @*<li>@Html.ActionLink("About", "About", "Home")</li>
                            <li>@Html.ActionLink("Contact", "Contact", "Home")</li>*@
                            <li>@Html.ActionLink("Movies", "Index", "Movies")</li>

                        </ul>
                    </div>
                    <ul><li class="nav navbar-nav">@Html.Partial("_LoginPartial")</li></ul>
                </div>
```

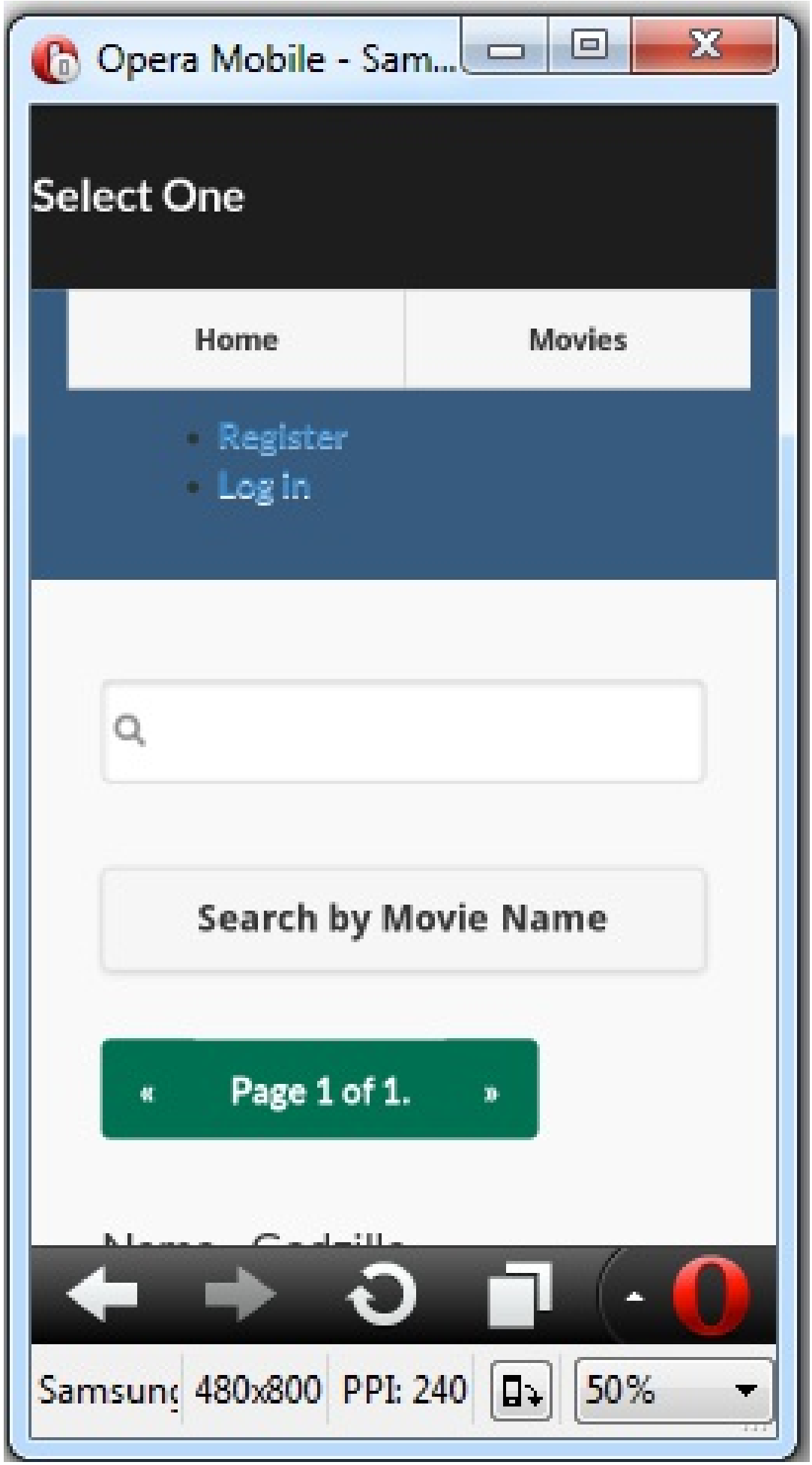
Then, it will appear as shown below


Opera Mobile - Sam...
Select One
Home
Movies
Register
Log in
Search by Movie Name
« Page 1 of 1. »
Name - Godzilla
Samsung
480x800
PPI: 240
50%

So, now it looks better, also header section and its text are nicely visible. So, if you see the content section that is not quite mobile optimized, it could have been bit better. So, let's fix this as well. So, what I do now, I'll include data-role content in my render body section as shown below.

```
<div>
    <section class="main-content" data-role="content">
        @RenderBody()
    </section>
</div>
```

Now, when I refresh the page, it will appear in a nicer way.


Opera Mobile - Sam...
Select One
Home
Movies
Register
Log in
Search by Movie Name
« Page 1 of 1. »
Samsung
480x800
PPI: 240
50%

Now, here one more important thing to understand, let's suppose you are writing some script which you want to invoke every time on each page load, then in that case you can't use "***document.ready()***" event in here, this will work only 1st time when your application loads. So, for this to work on every page there is a different bind event known as "***pageinit***", which will execute the script on every page rather every link you click. Below, is the snapshot of pageinit event.

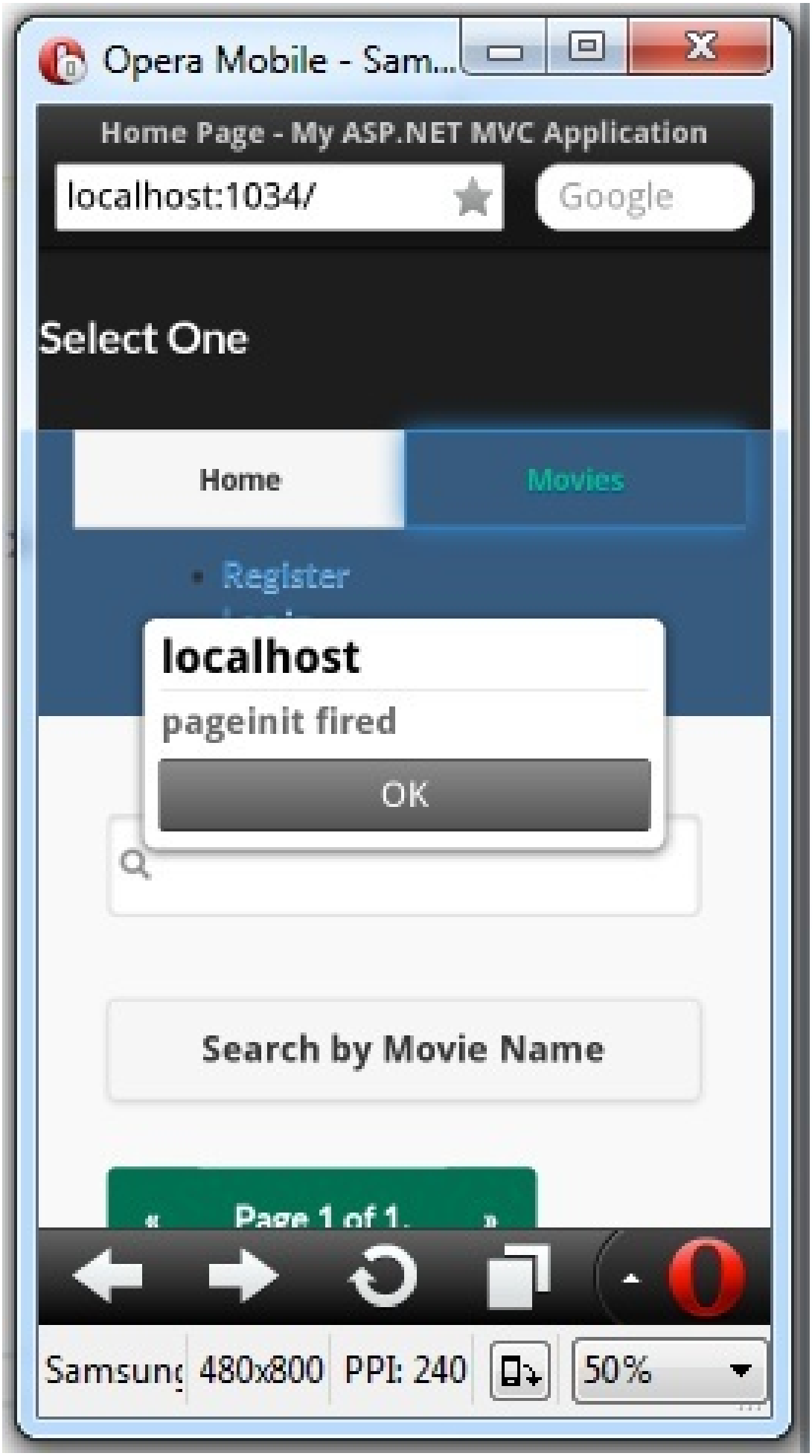
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>@ViewBag.Title - My ASP.NET MVC Application</title>
    <link href="~/favicon.ico" rel="shortcut icon" type="image/x-icon" />
    <meta name="viewport" content="width=device-width,initial-scale=1.0" />

    <link href="~/Content/jquery.mobile-1.4.2.css" rel="stylesheet" />
    <script src="~/Scripts/jquery-2.1.0.min.js"></script>

    <link href="~/Content/BootStrap-theme-movie.css" rel="stylesheet" />
    <script src="~/Scripts/jquery.mobile-1.4.2.js"></script>
    @*@Styles.Render("~/Content/css")*@
    @Scripts.Render("~/bundles/modernizr")

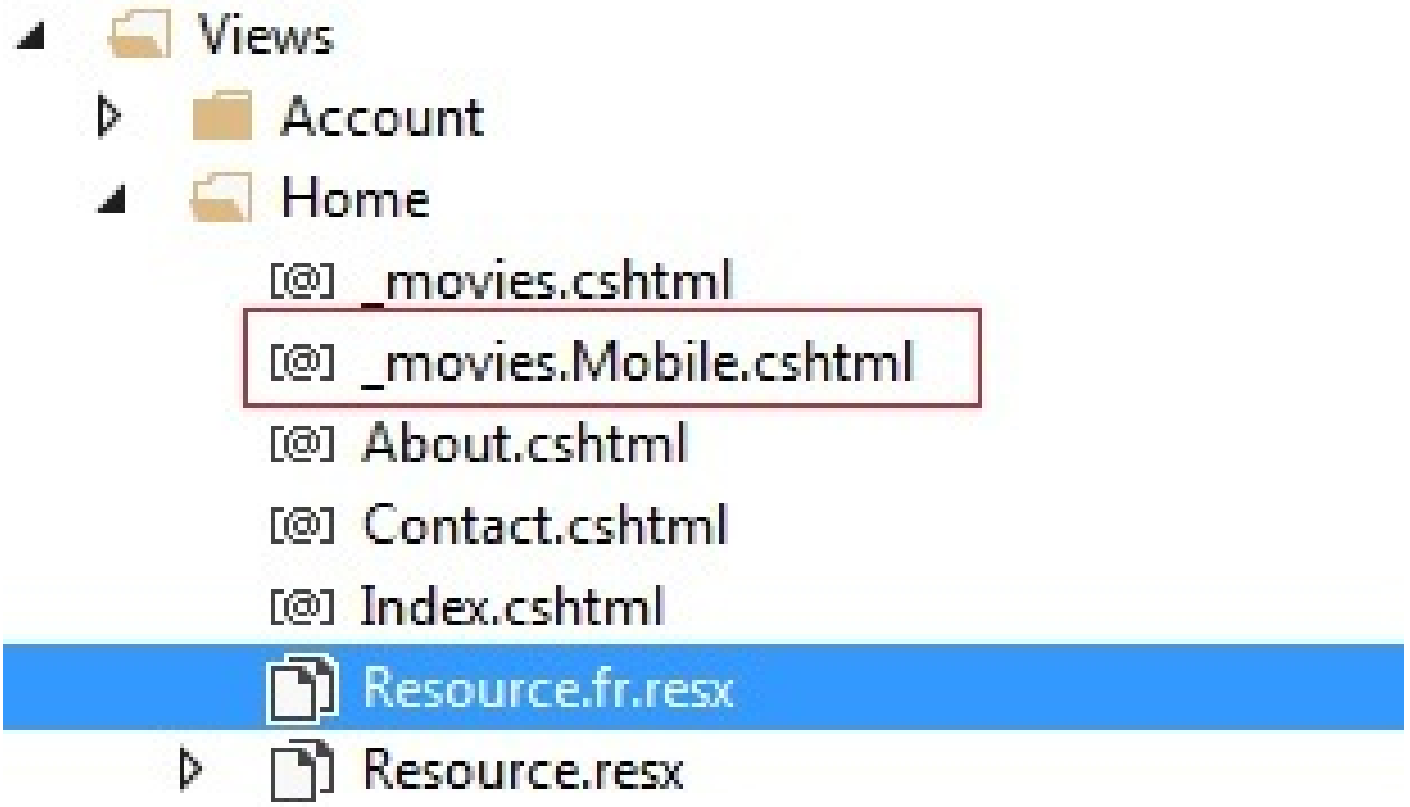
    <script>
        $(document).bind("pageinit", function() {
            alert("pageinit fired");
        });
    </script>
</head>
```

Now, with that change in place, when you refresh the page, it will produce the alert as shown below.


Opera Mobile - Sam...
Home Page - My ASP.NET MVC Application
localhost:1034/
Google
Select One
Home
Movies
Register
localhost
pageinit fired
OK
Search by Movie Name
« Page 1 of 1. »
Samsung
480x800
PPI: 240
50%

Collapsibles:-

Now, in this section we'll see that how to work with collapsibles in our mobile browser. Currently, in our home page, we need to drag completely to see the contents below. But, we can optimize this behavior on the home page by using collapsible wizard. And for using the same, it means I need to change the partial view, so I'll include one new page which is basically mobile version of the partial view page as shown below in the screen shot.

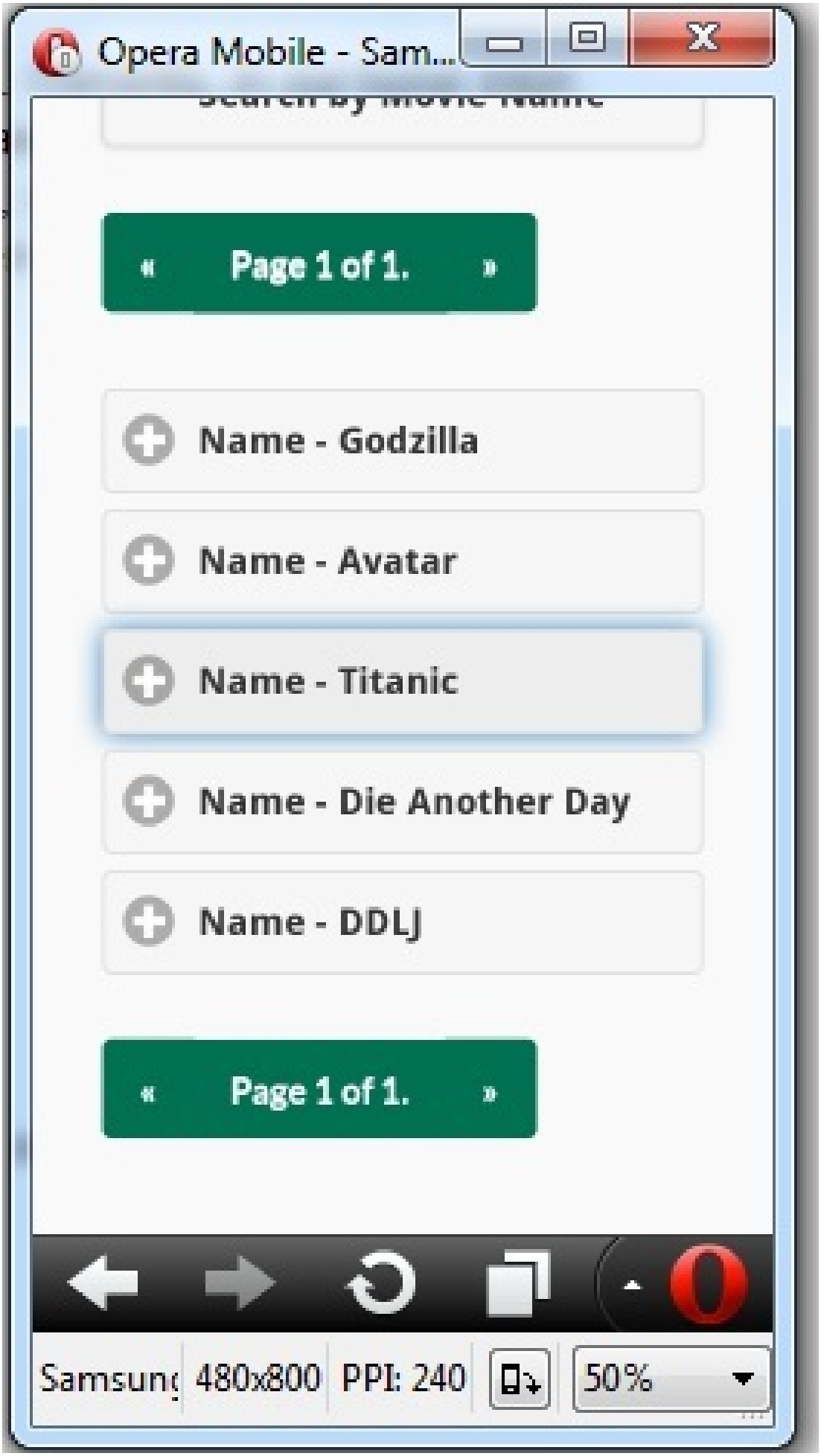


Now, in the page I have just added the data role for collapsible as you can see below in the screen shot.

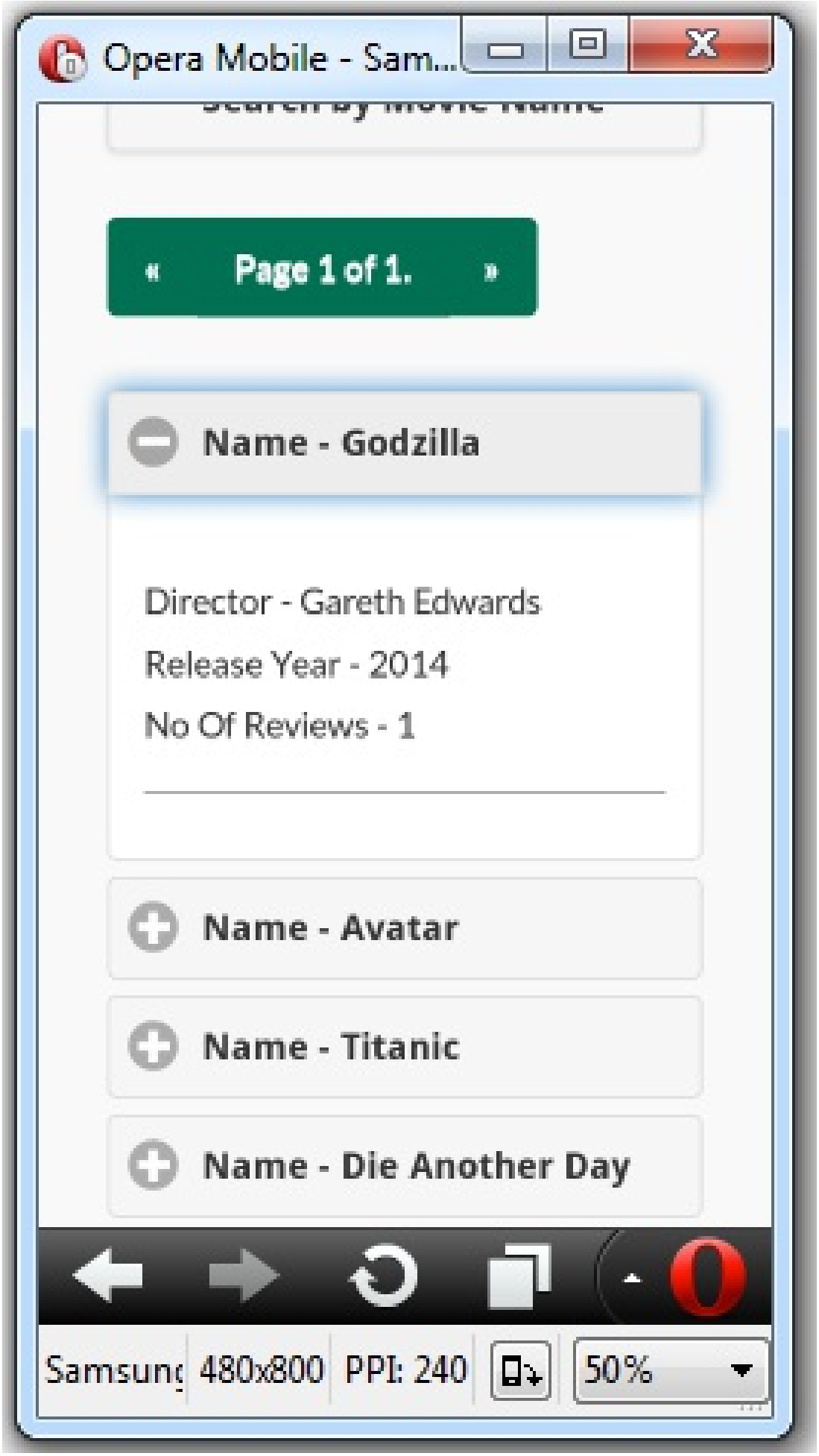
_movies.Mobile.cshtml | _movies.cshtml | _Layout.Mobile.cshtml | _Layout.cshtml

```
1  @model IPagedList<MovieReview.Models.MovieViewModel>
2
3
4  <div id="movieDiv">
5
6      <div class="pagedList" data-movie-target="#movieDiv">
7          @Html.PagedListPager(Model, page => Url.Action("Index", new { page}),
8          PagedListRenderOptions.MinimalWithPageCountText)
9      </div>
10         @foreach (var item in Model) {
11         <div data-role="collapsible">
12             <h4>Name - @item.MovieName</h4><br />
13             <h5>Director - @item.DirectorName</h5>
14             <h5>Release Year - @item.ReleaseYear</h5>
15             <h5>No Of Reviews - @item.NoOfReviews</h5>
16             <hr />
17         </div>
18 }
19     <div class="pagedList" data-movie-target="#movieDiv">
20         @Html.PagedListPager(Model, page => Url.Action("Index", new { page }),
21         PagedListRenderOptions.MinimalWithPageCountText)
22     </div>
23 </div>
```

So, with the above change in place, when I go ahead and refresh my page in mobile browser, it will produce the below result.


Opera Mobile - Sam...
« Page 1 of 1. »
Name - Godzilla
Name - Avatar
Name - Titanic
Name - Die Another Day
Name - DDLJ
« Page 1 of 1. »
Samsung
480x800
PPI: 240
50%

Now, when you click on any of these collapsible links, it will open like below having the complete movie details.


Opera Mobile - Sam...
« Page 1 of 1. »
Name - Godzilla
Director - Gareth Edwards
Release Year - 2014
No Of Reviews - 1
Name - Avatar
Name - Titanic
Name - Die Another Day
Samsung
480x800
PPI: 240
50%

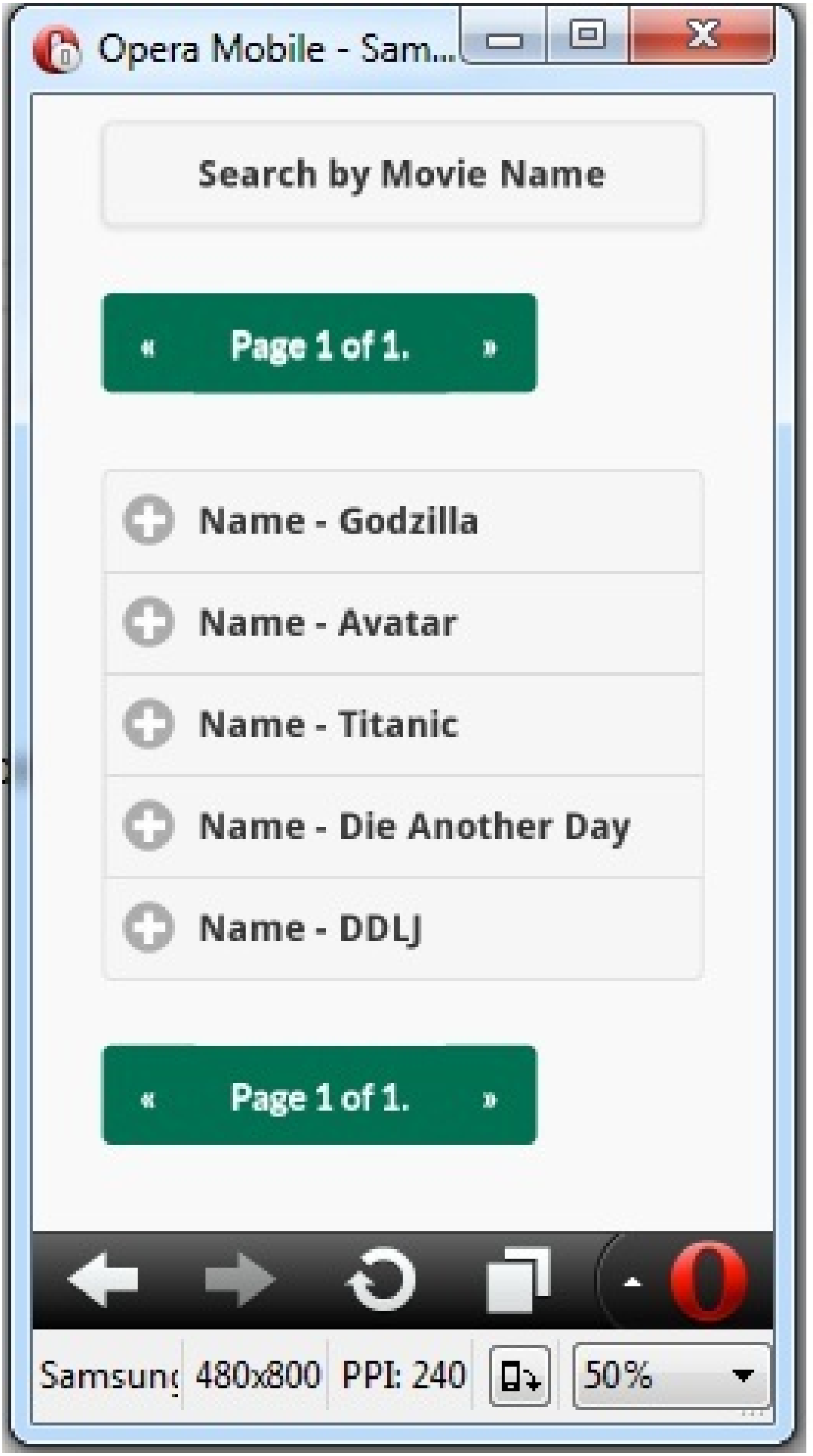
Now, you can improvise this bit more by trimming the space between the collapsibles. So, for this I have used widget **collapsible-set** as shown below in the screen shot.

_movies.Mobile.cshtml | _movies.cshtml | _Layout.Mobile.cshtml | _Layout.cshtml

```
1  @model IPagedList<MovieReview.Models.MovieViewModel>
2
3
4  <div id="movieDiv">
5
6      <div class="pagedList" data-movie-target="#movieDiv">
7          @Html.PagedListPager(Model, page => Url.Action("Index", new { page }),
8              PagedListRenderOptions.MinimalWithPageCountText)
9      </div>
10     <div data-role="collapsible-set">
11         @foreach (var item in Model)
12         {
13             <div data-role="collapsible">
14                 <h4>Name - @item.MovieName</h4><br />
15                 <h5>Director - @item.DirectorName</h5>
16                 <h5>Release Year - @item.ReleaseYear</h5>
17                 <h5>No Of Reviews - @item.NoOfReviews</h5>
18                 <hr />
19             </div>
20         }
21     </div>
22     <div class="pagedList" data-movie-target="#movieDiv">
23         @Html.PagedListPager(Model, page => Url.Action("Index", new { page }),
24         PagedListRenderOptions.MinimalWithPageCountText)
25     </div>
26 </div>
```

Now, with the above change in place, it

will trim the space in between collapsibles and produce it like a set as shown below.


Opera Mobile - Sam...
Search by Movie Name
« Page 1 of 1. »
Name - Godzilla
Name - Avatar
Name - Titanic
Name - Die Another Day
Name - DDLJ
« Page 1 of 1. »
Samsung
480x800
PPI: 240
50%

Now, one more important thing to note here that by default ajax features are on for all navigations links you click, so because of this sometimes by mobile browser doesn't behave or doesn't load the complete page, it just renders what is being loaded earlier. So, to fix this behavior we need to tell the mobile browser, please switch off the Ajax enabled feature. And we need to pass this instruction before Mobile jQuery script gets loaded as shown below.

_movies.Mobile.cshtml | _movies.cshtml | _Layout.Mobile.cshtml | _Layout.cshtml

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>@ViewBag.Title - My ASP.NET MVC Application</title>
    <link href="~/favicon.ico" rel="shortcut icon" type="image/x-icon" />
    <meta name="viewport" content="width=device-width,initial-scale=1.0" />

    <link href="~/Content/jquery.mobile-1.4.2.css" rel="stylesheet" />
    <script src="~/Scripts/jquery-2.1.0.min.js"></script>

    <link href="~/Content/BootStrap-theme-movie.css" rel="stylesheet" />
    <script>
        $(document).bind("mobileinit", function() {
            $.mobile.ajaxEnabled = false;
        });
    </script>
    <script src="~/Scripts/jquery.mobile-1.4.2.js"></script>
    @*@Styles.Render("~/Content/css")*@
    @Scripts.Render("~/bundles/modernizr")


</head>

```

Now, when you test your app in mobile, it will behave same way as it is behaving in desktop browser.

Grids and Buttons:-

Now, one more change which you can do here is aligning the content in grid format. Now, for using grid format, we

have class with different specifications say "***a,b,c,d***" which represents "***2,3,4 or 5***" columns. So, below in the code I have just used three columns set and surrounded the same with individual divs.

_movies.Mobile.cshtml | _movies.cshtml | _Layout.Mobile.cshtml | _Layout.cshtml

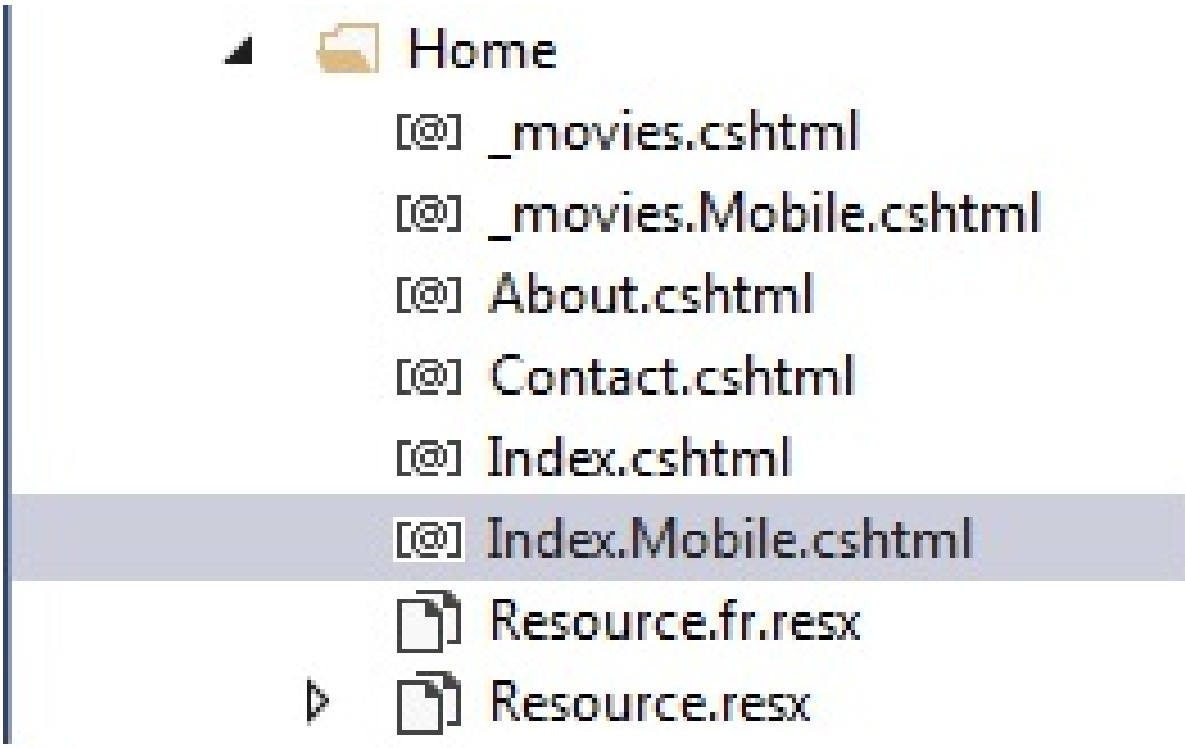
```
@model IPagedList<MovieReview.Models.MovieViewModel>


<div id="movieDiv">

    <div class="pagedList" data-movie-target="#movieDiv">
        @Html.PagedListPager(Model, page => Url.Action("Index", new { page }),
        PagedListRenderOptions.MinimalWithPageCountText)
    </div>
    @foreach (var item in Model)
    {
        <div class="ui-grid-b">
            <div class="ui-block-a">
                <h4>Name - @item.MovieName</h4><br />
            </div>
            <div class="ui-block-b">
                <div> <h5>Director - @item.DirectorName</h5></div>
                <div>
                    <h5>Release Year - @item.ReleaseYear</h5>
                </div>
                <div>
                    <h5>No Of Reviews - @item.NoOfReviews</h5>
                </div>
            </div>
            <hr />
        </div>
    }
    <div class="pagedList" data-movie-target="#movieDiv">
        @Html.PagedListPager(Model, page => Url.Action("Index", new { page }),
        PagedListRenderOptions.MinimalWithPageCountText)
    </div>
</div>
```

Now, "***ui-grid***" here is initializing the grid and blocks are representing which item should be taken as 1st or 2nd or subsequent.

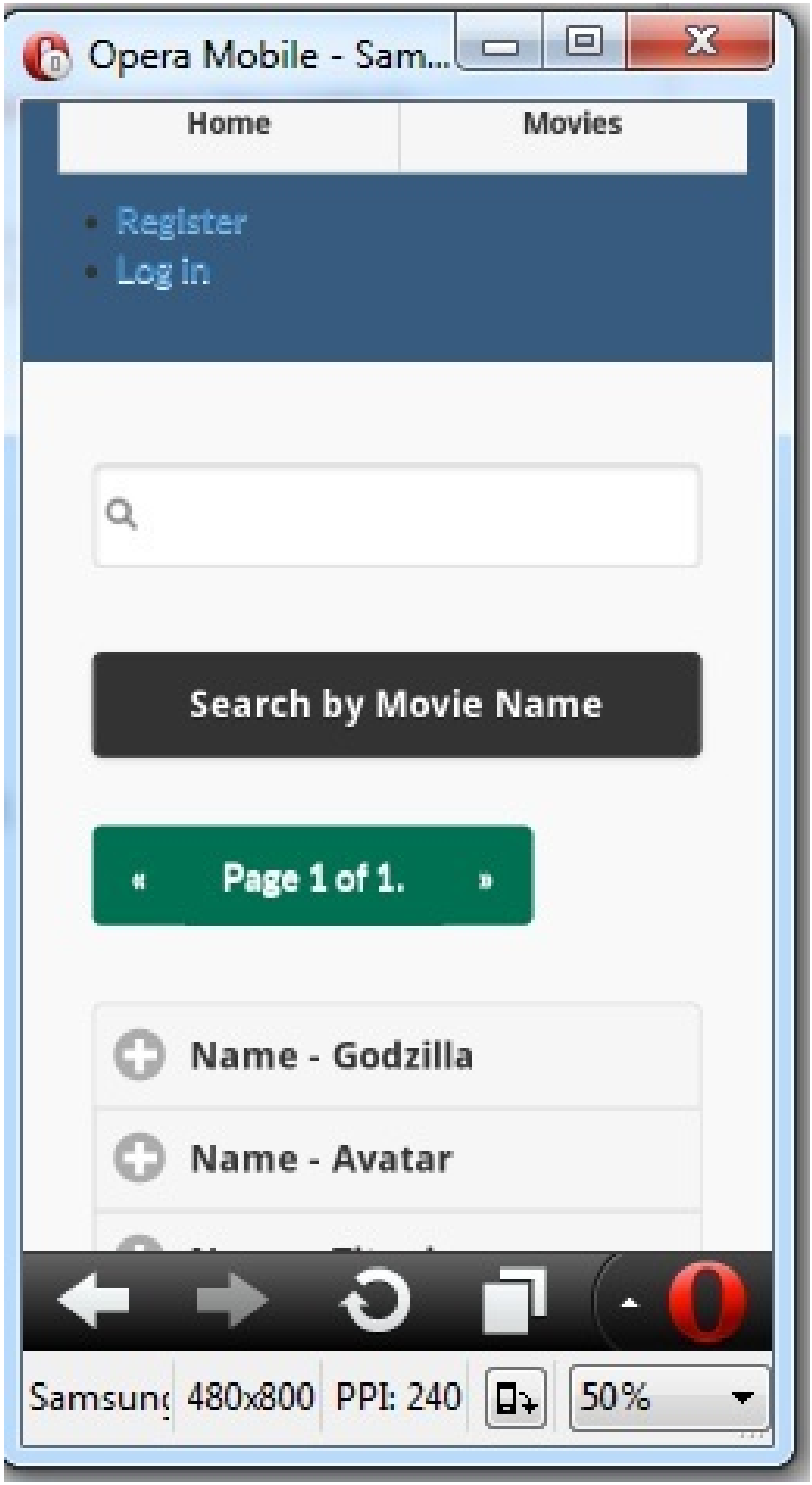
Now, the next thing which we can improvise is button look and feel to make it more mobile friendly. So, now for doing this I need to include one more mobile version of index page so that I can change the setting there.



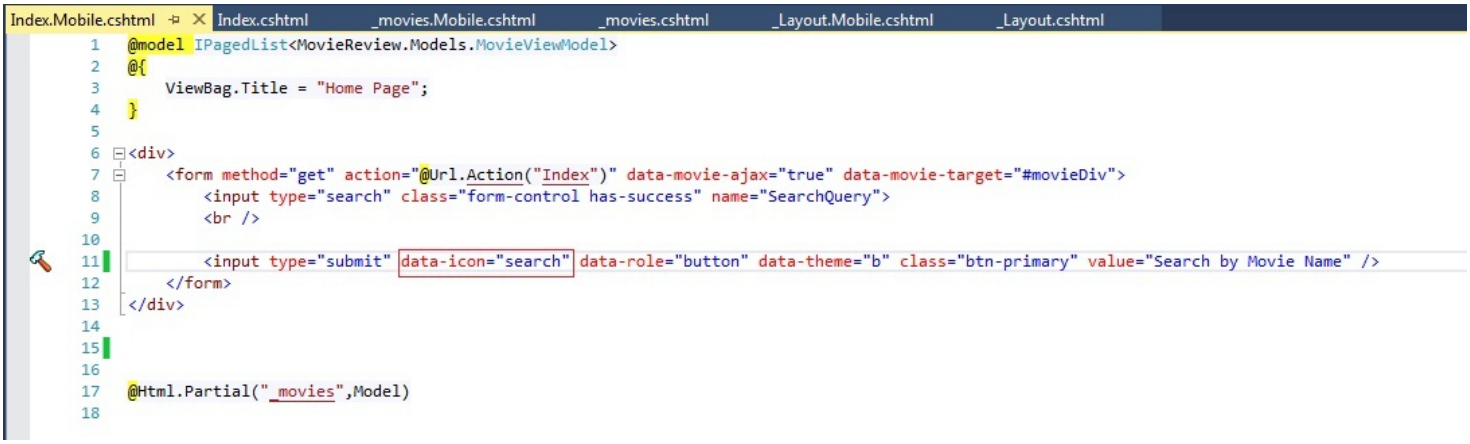
Now, in here I have assigned button, with role button and theme b, just to give it a bit contrast.



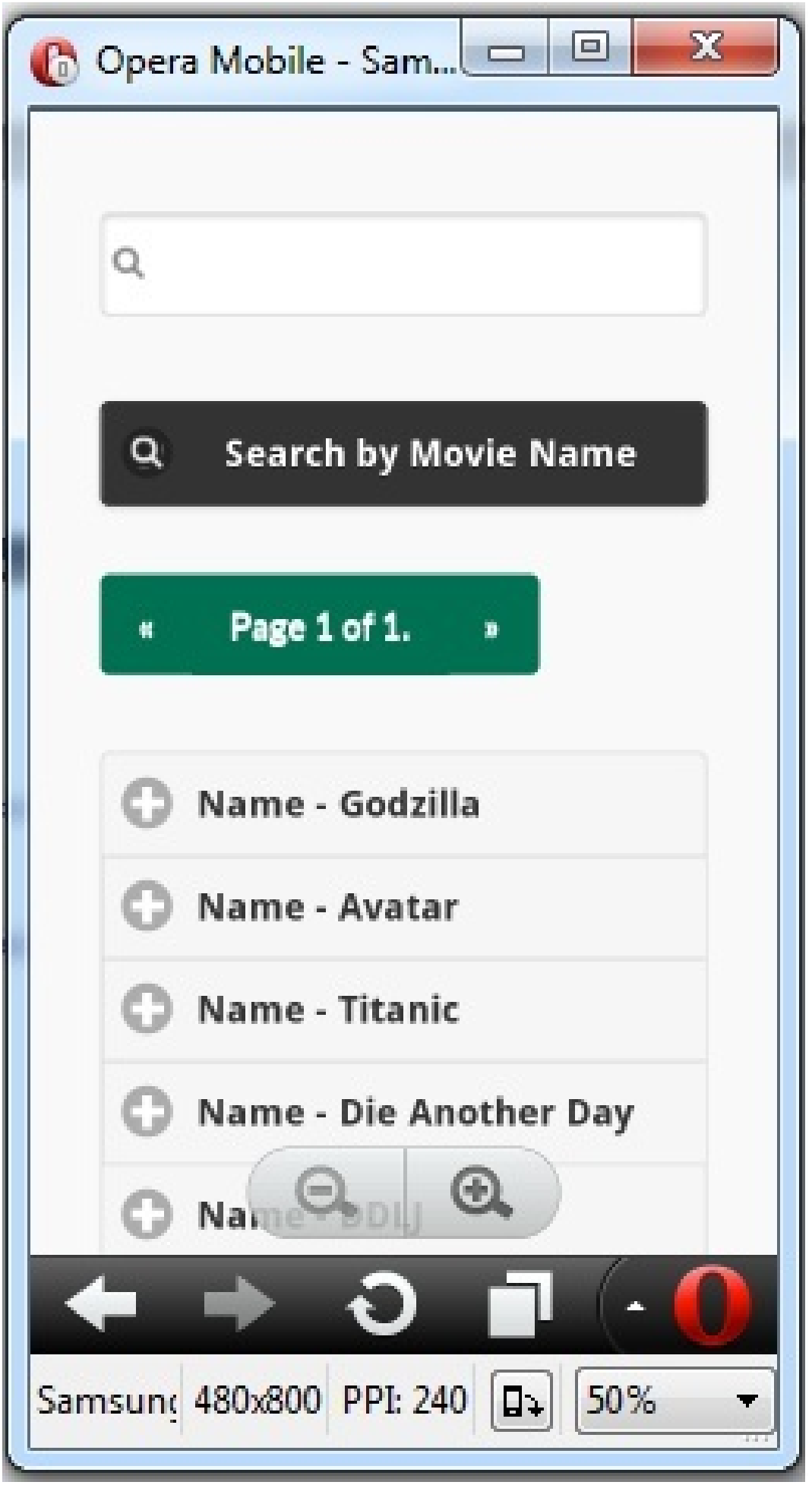
So, now when I save the same and refresh my mobile browser, it will take the effect.


Opera Mobile - Sam...
Home
Movies
Register
Log in
Search by Movie Name
« Page 1 of 1. »
Name - Godzilla
Name - Avatar
Samsung
480x800
PPI: 240
50%

Now, buttons can also have icons, so I can use search icon as shown below.

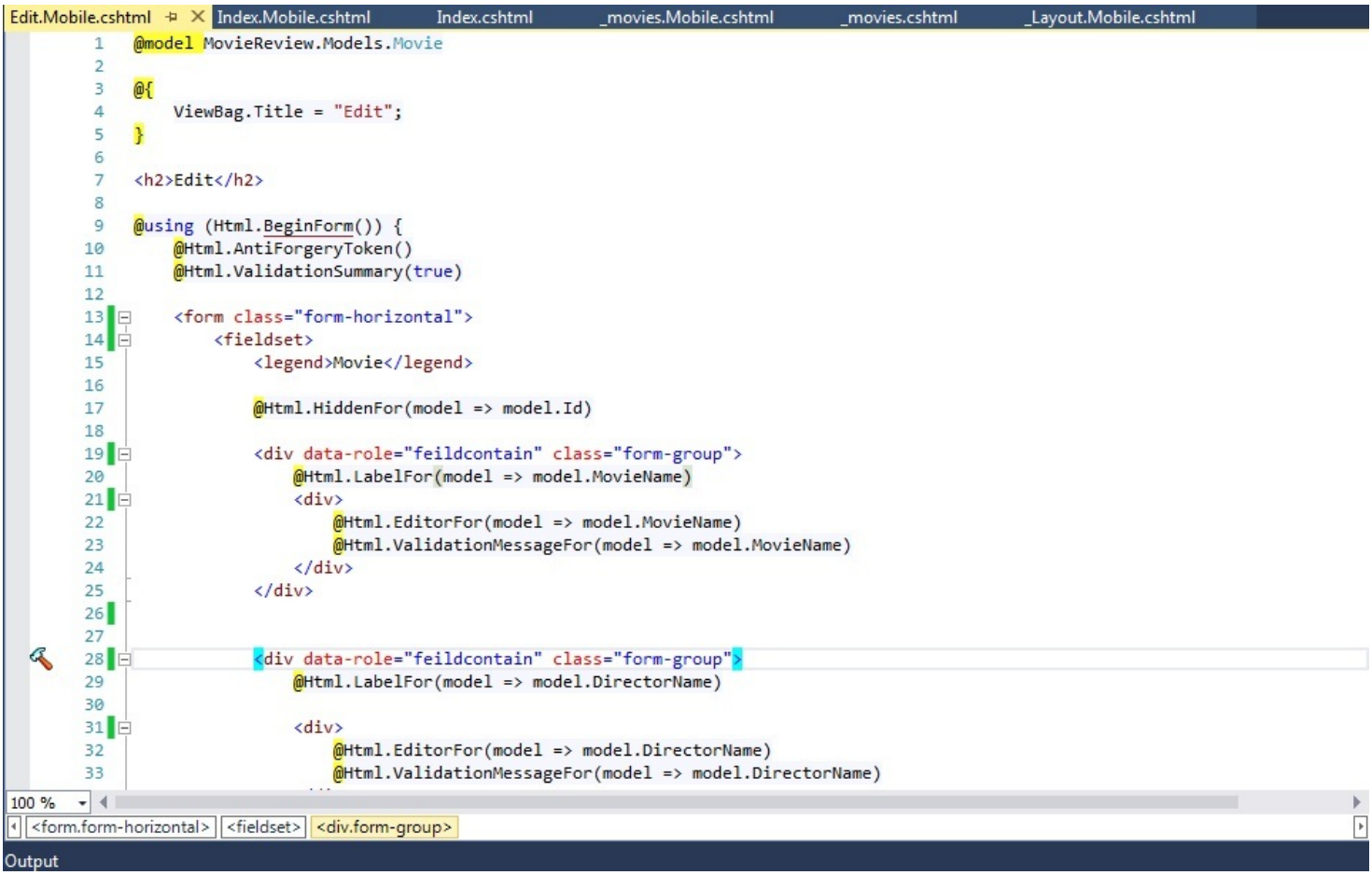


With this it will produce the below output.

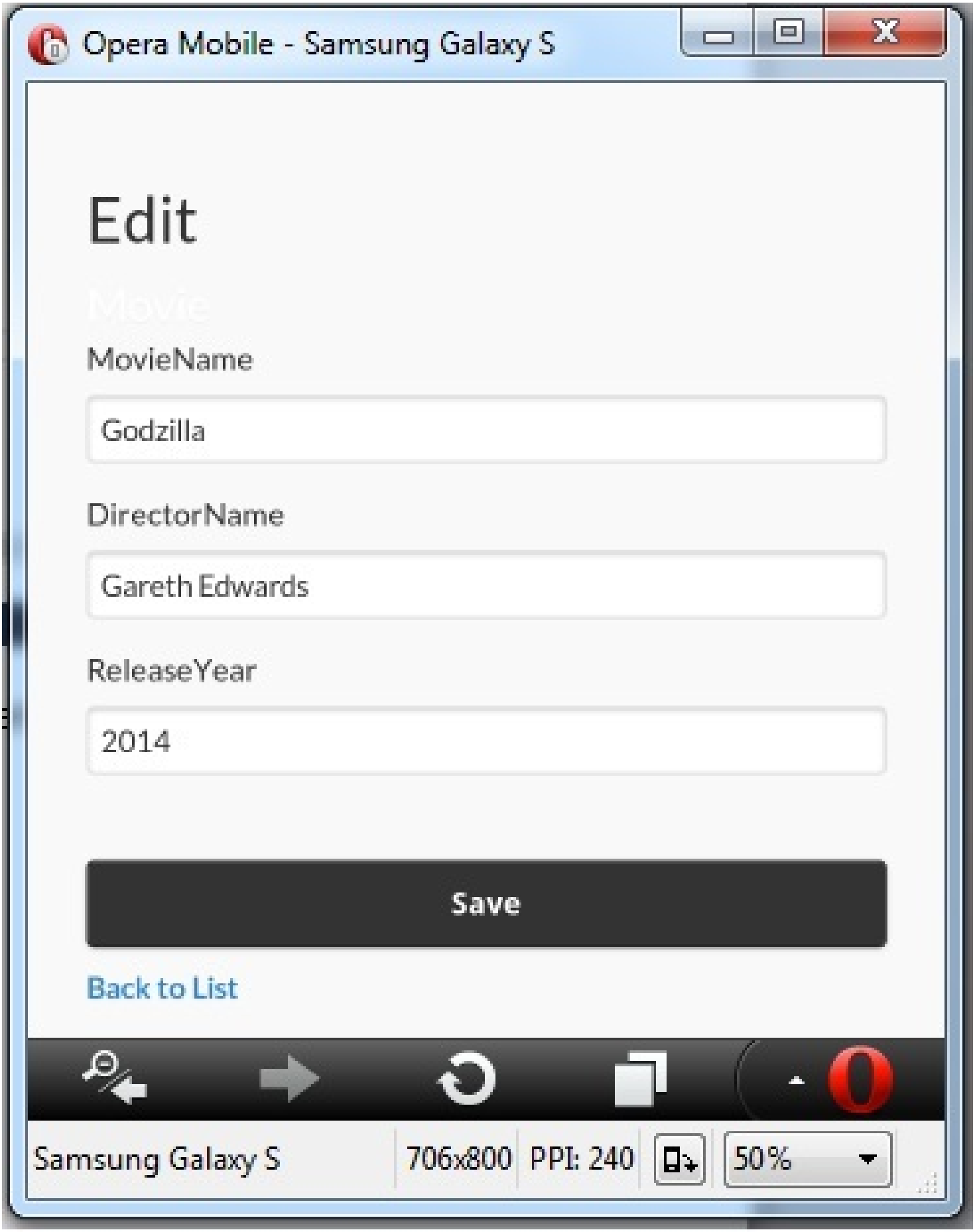

Opera Mobile - Sam...
Search by Movie Name
« Page 1 of 1. »
Name - Godzilla
Name - Avatar
Name - Titanic
Name - Die Another Day
Samsung
480x800
PPI: 240
50%

Forms:-

Now, in this section, we'll look more about Forms or Friendly Forms, which not only looks better, but better responsive and touch sensitive. So, here I am going to use "***data-role="feildcontain""***". Now, "***fieldcontain***" is the one which aligns the markup and present the text in nicer way. So, we'll start doing the forms implementation from the Edit page.



Now, I have also setup the theme for the button. So, when you refresh the page, you will get to see nice rounded corner areas with themed button.


Opera Mobile - Samsung Galaxy S
Edit
Movie
MovieName
Godzilla
DirectorName
Gareth Edwards
ReleaseYear
2014
Save
Back to List
Samsung Galaxy S
706x800
PPI: 240
50%

So, obviously there are many settings which you can think off and apply the same in your application as per your requirement. You can check many more widgets on the JQuery Mobile Site with example.

Summary:-

So, in this section we have seen many things; like we started with the basics of JQM. Then we took our Movie Review site and converted the same to support Mobile browsers as well. For doing this, we have come across many widgets and settings. With these many settings, we have made our app mobile friendly.

Obviously, whatever things which we have seen in this section, JQM is not only limited to this. There are many more things which you can experiment with when you look at the JQM site.

Chapter 14: ASP.NET MVC 5

WHAT DO you find in this CHAPTER?

⇒ Introduction

⇒ Scaffolding

⇒ ASP.Net Identity

⇒ One ASP.Net

⇒ Bootstrap

⇒ Attribute Routing

⇒ Filter Overrides

⇒ Summary

Introduction:-

Hi Friends, now we have come to a section where in we can discuss MVC 5 Features. In this section, I'll be doing basic walkthrough with the above listed topics. So, basically these topics are the

extension of what we built so far. The ASP.NET MVC 5 Framework is the latest update to Microsoft's popular ASP.NET web platform. It provides an Extensible, High-Quality programming model which allows you to build Dynamic, Data-driven websites, focusing on a cleaner architecture and Test-driven development.

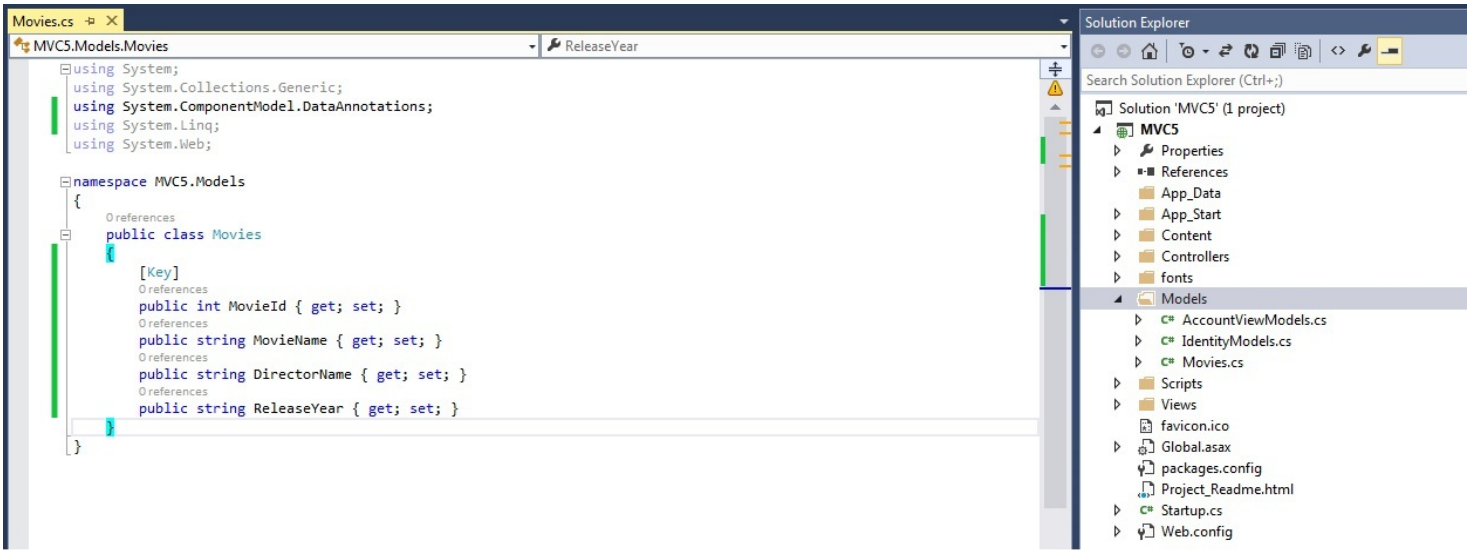
ASP.NET MVC 5 contains a number of improvements over previous versions, including some new features, improved user experiences; native support for JavaScript libraries to build multi-platform CSS and HTML 5 enabled sites and better tooling support.

Scaffolding:-

So, Visual Studio 2013 includes a new Scaffolding Framework for ASP.NET MVC 5., Scaffolding is basically a code generation framework provided for ASP.NET Web Applications. Using this framework, we can easily generate code that interacts with our data models. Let's get started with demo in this section. Let's open VS 2013 and create one new project. Here, I have already created with the name MVC5 as shown below in the screen shot.

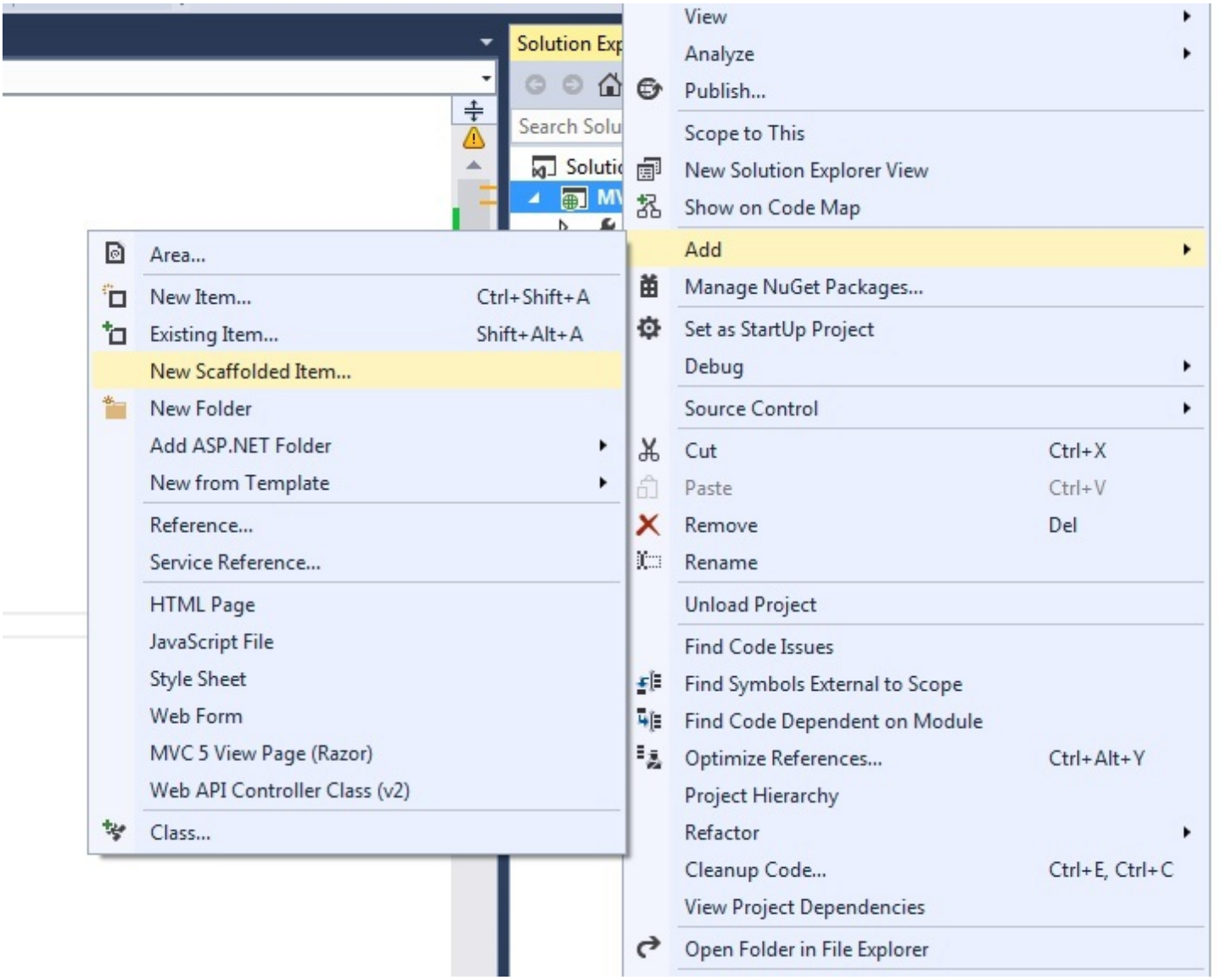


Now, let me go ahead and create one new class in the Models folder with the name Movies as shown below.



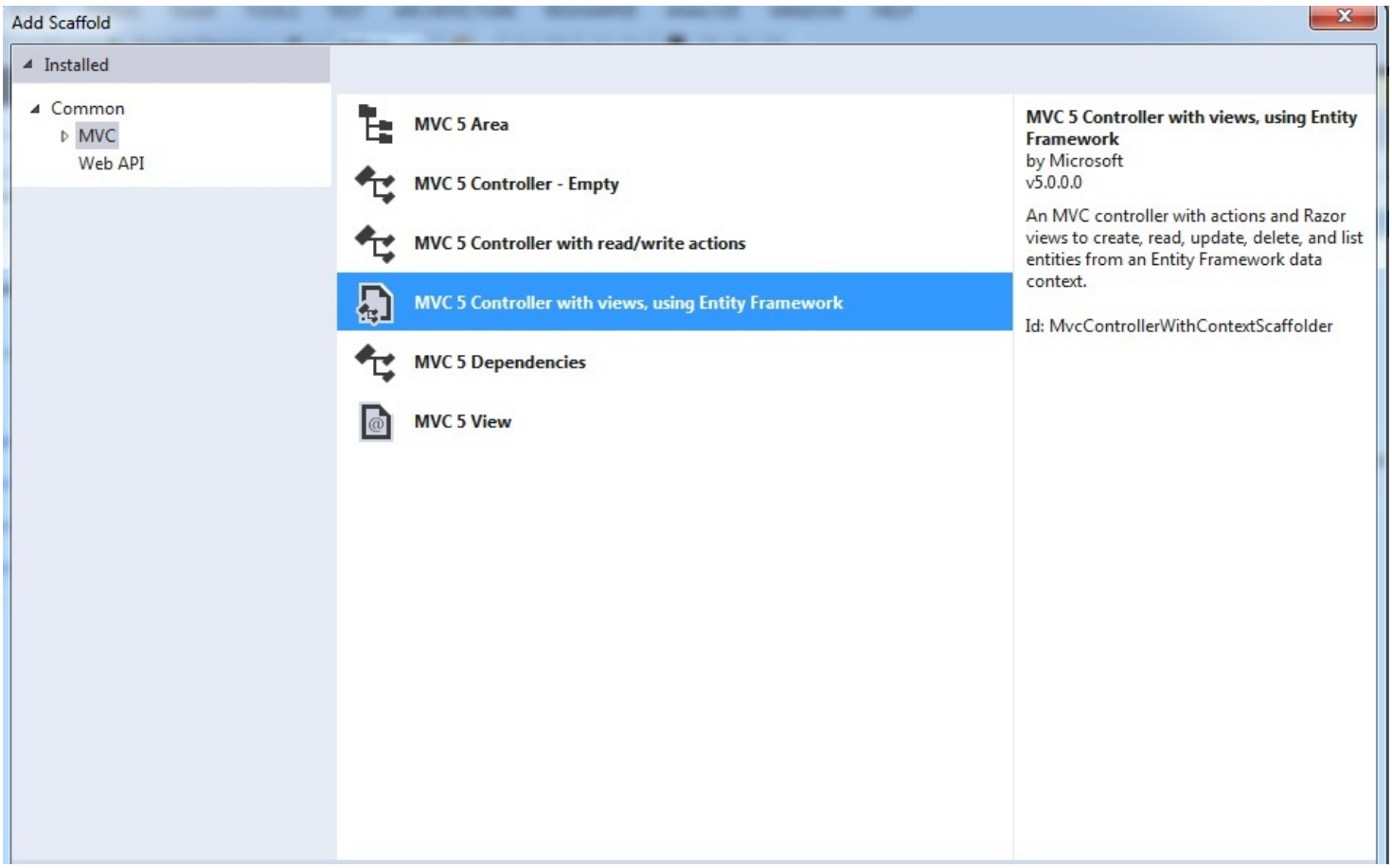
Now, in the project right-click and say

add "***New Scaffolded Item***" as shown below.

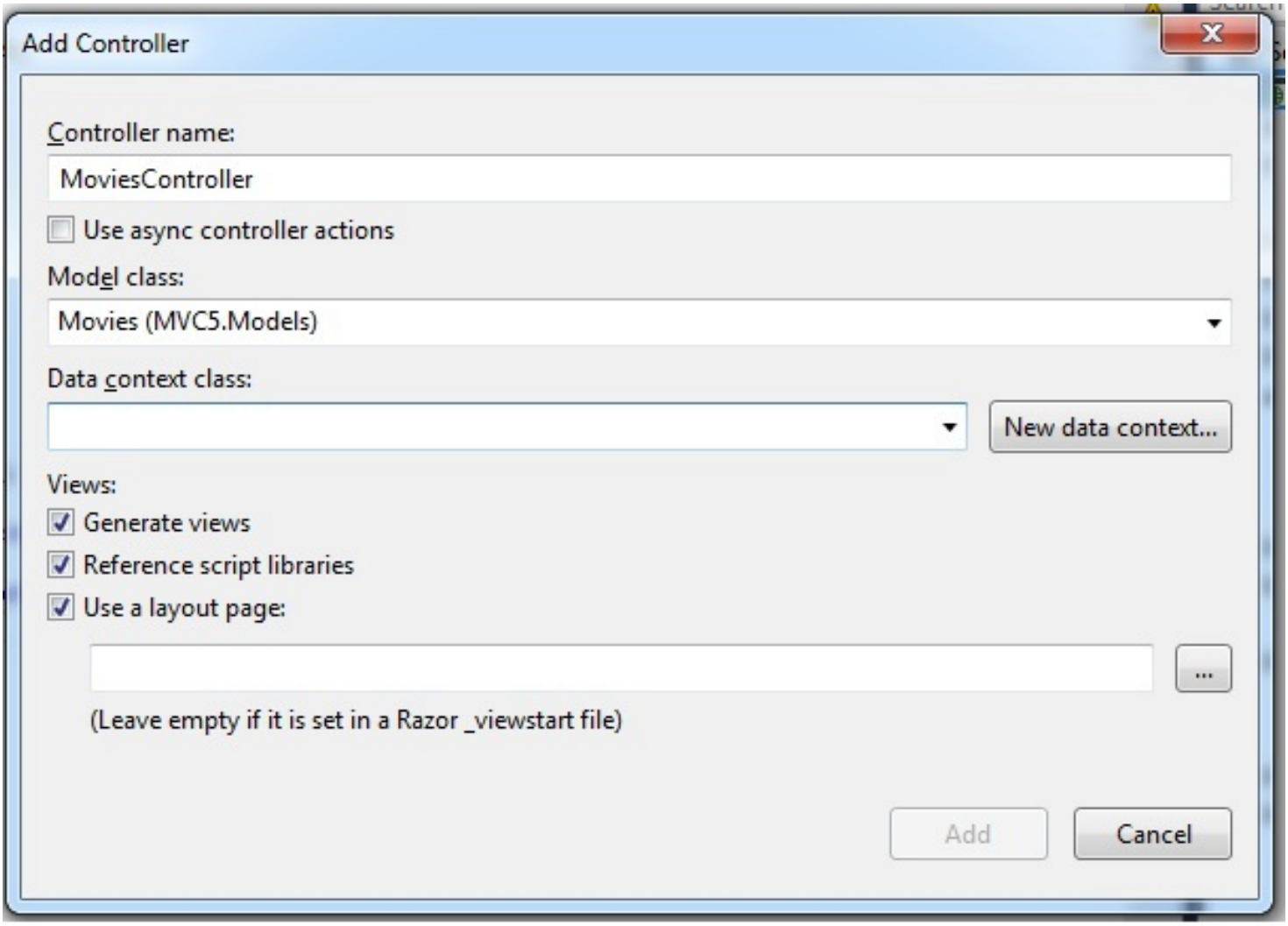


Then, a new window will get open which will have two options, one for MVC and another for Web API.

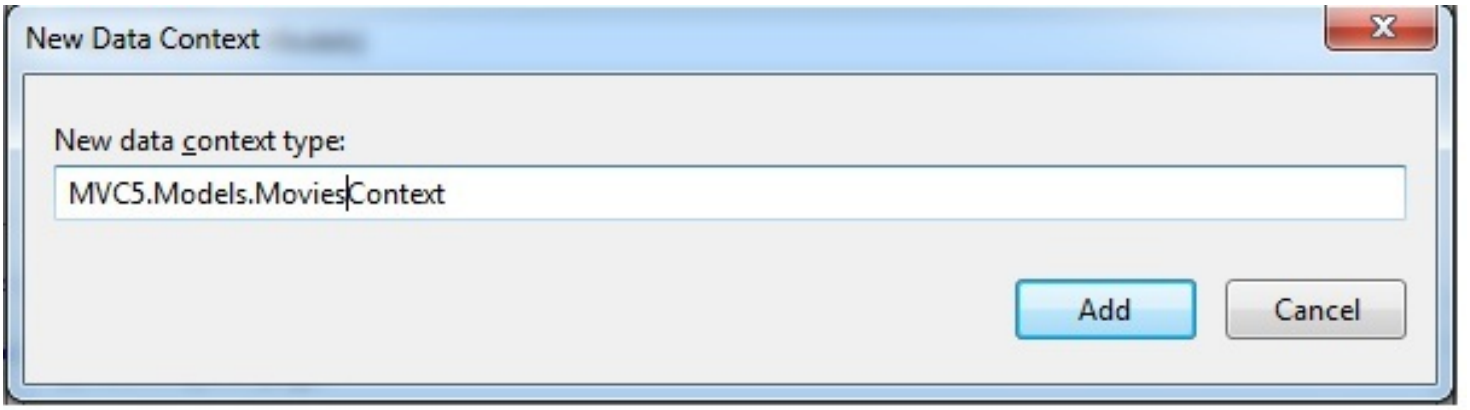
Remember, Web API 2 is also released with this version. But, we need to select MVC from there.



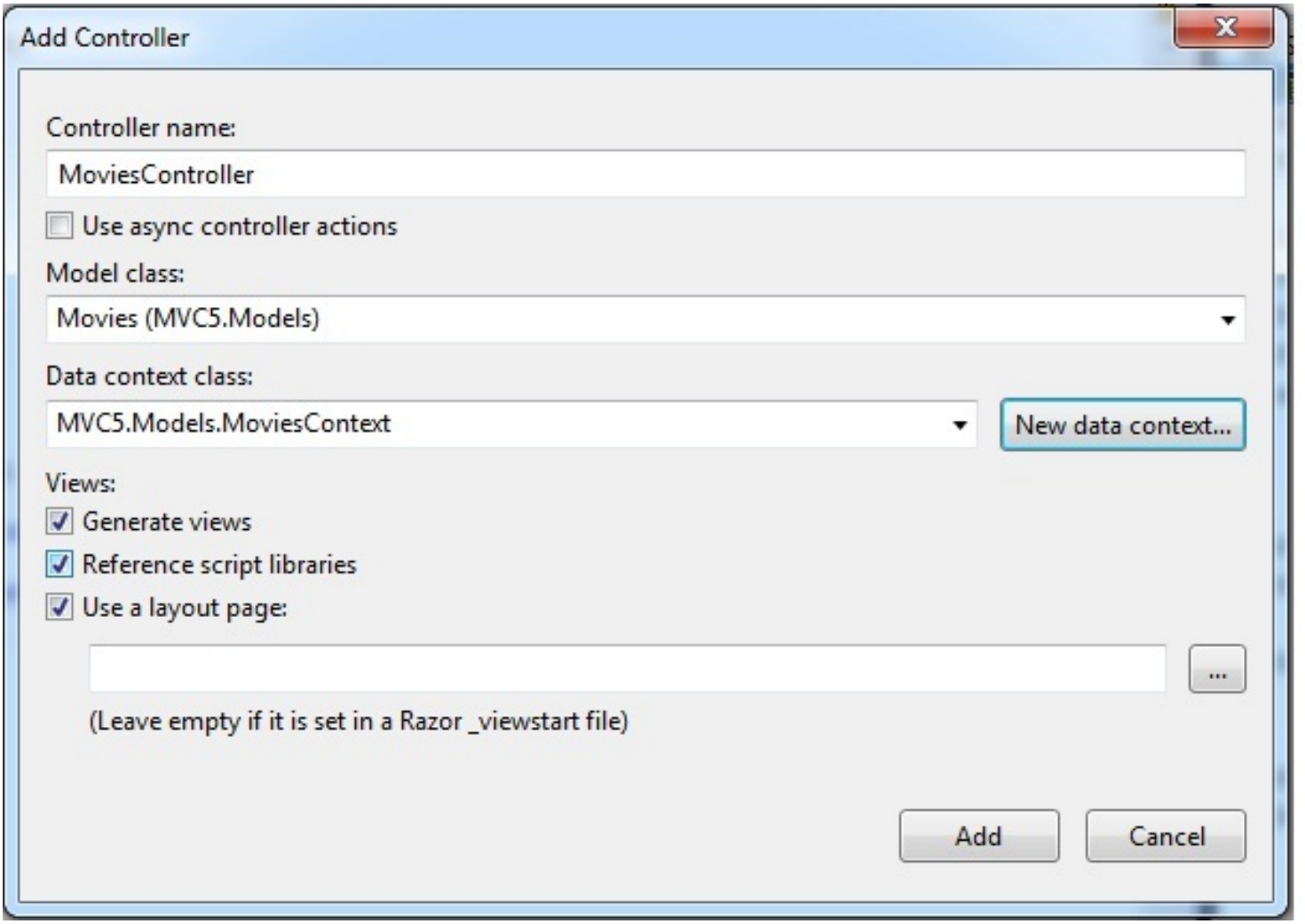
Then, click on Add button. Once, you click on Add button, below window will open



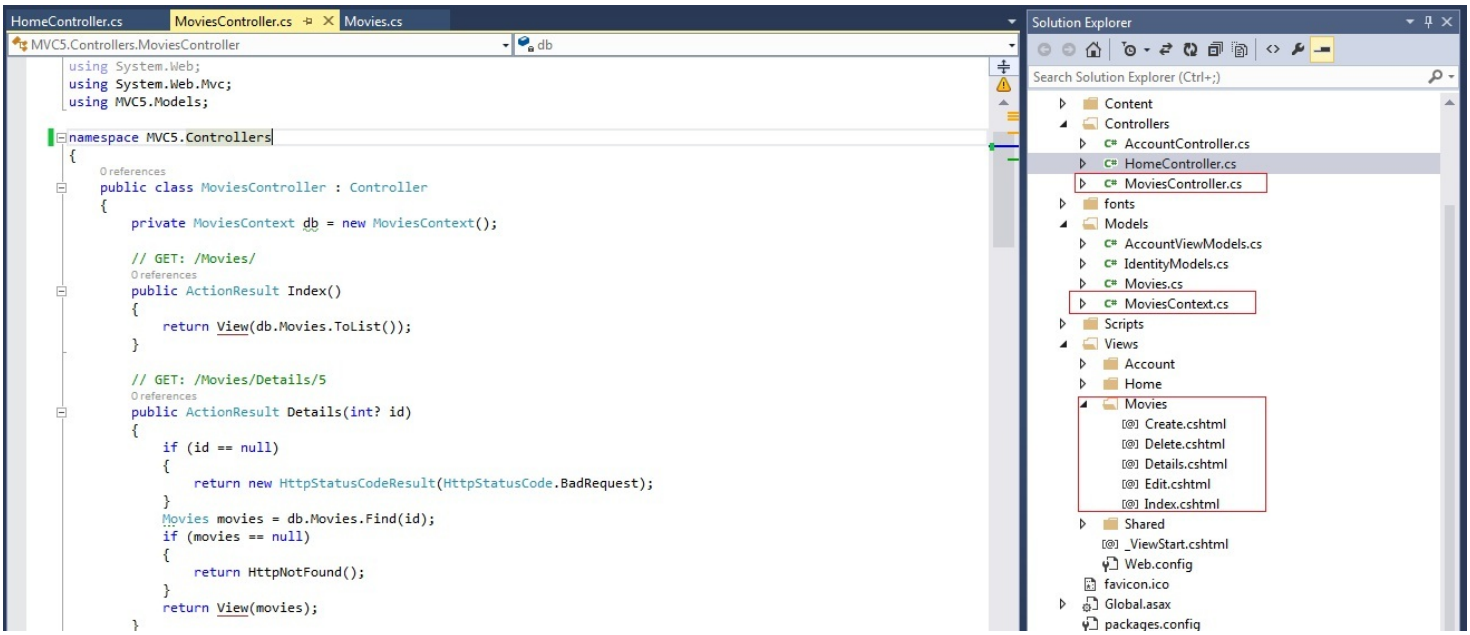
Now, here we need to create new context class as we don't have one. So, click on "***New data context***". This will open below window



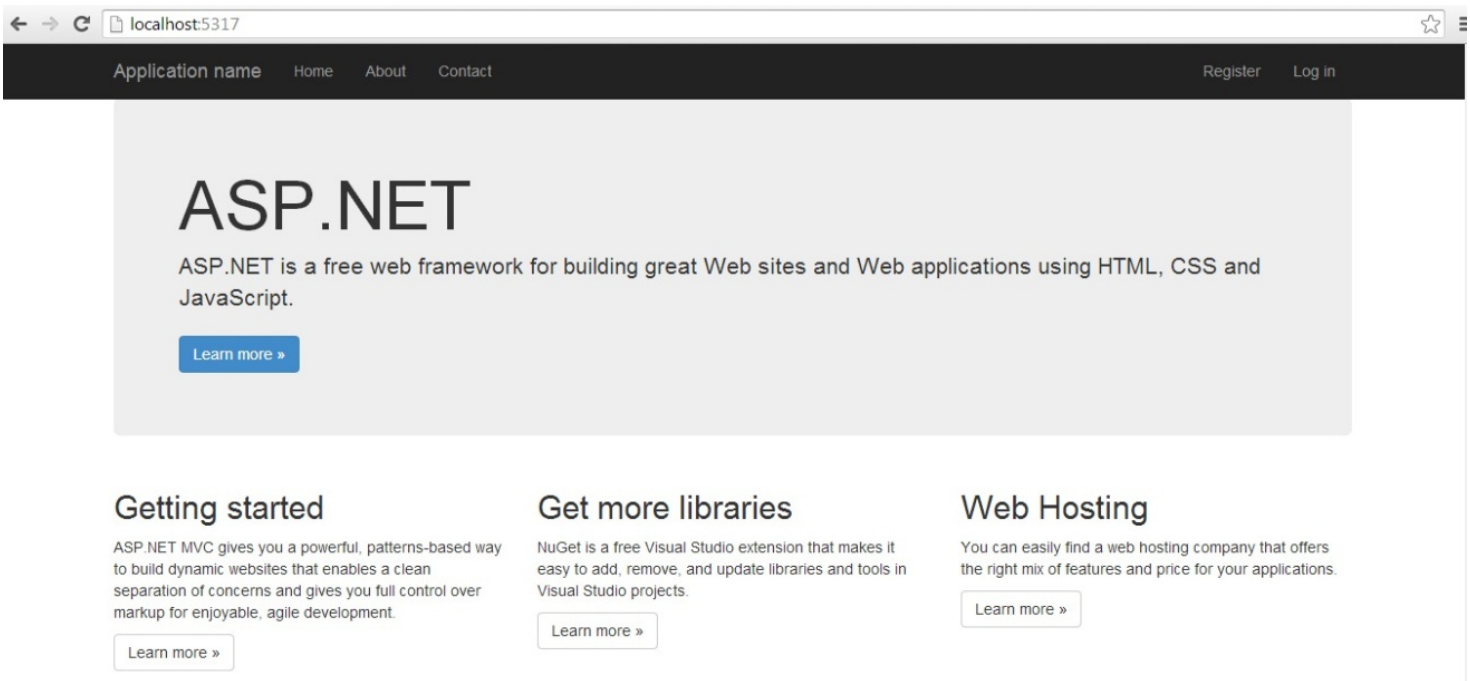
After entering the above values, just click on add, then it will go to the parent window with these context setting as shown below.



Now, click on Add, this will generate the required components as shown below.



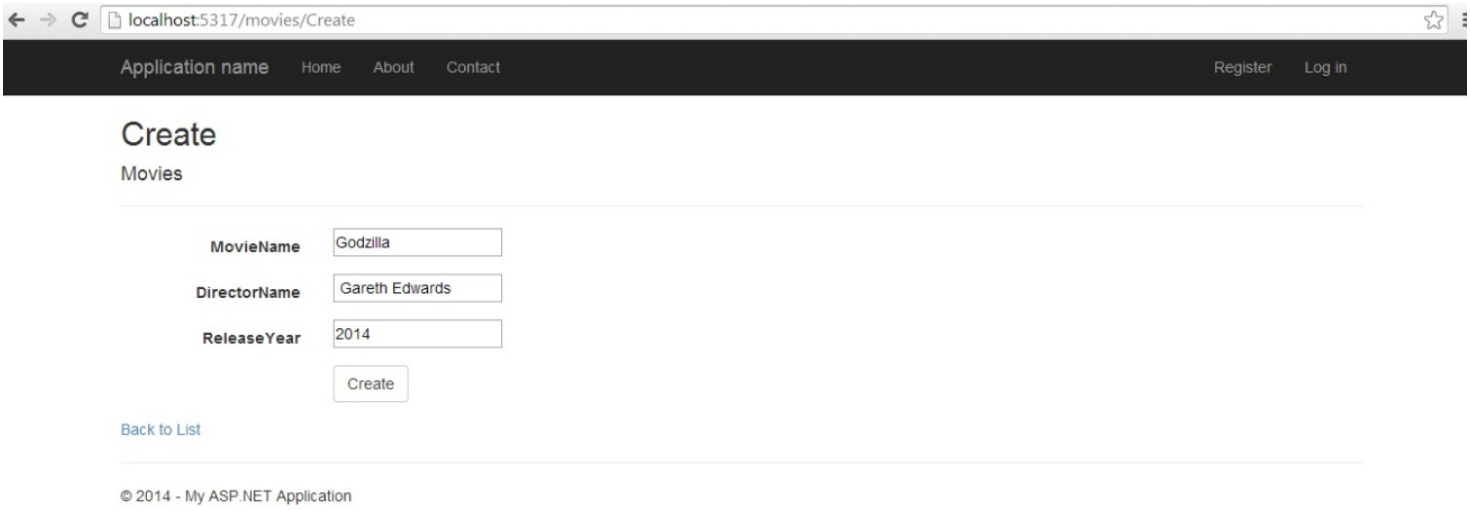
Now, when I build and refresh the page, it will produce the below page. So, below page is the default page of the MVC 5 application.



Now, when I navigate to the url "***http://localhost:5317/movies***", it will produce me the below output as shown below.

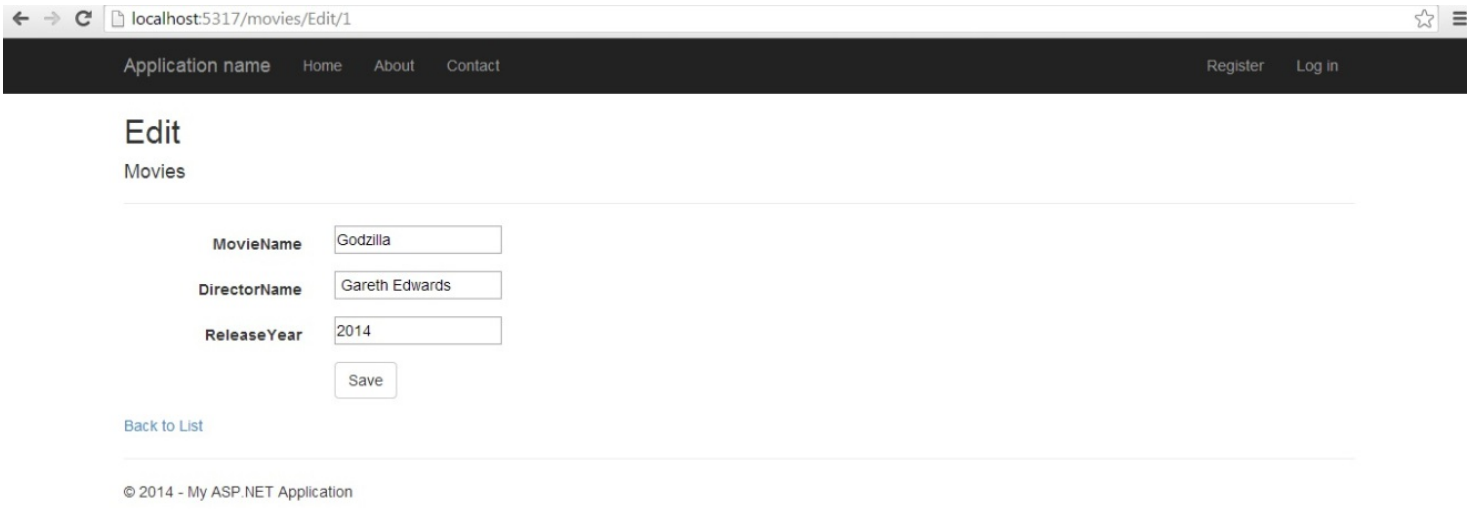


Now, remember that we haven't seeded any Test Data as we did while building the application. But, that's ok we can go ahead and create few records in it as shown below.

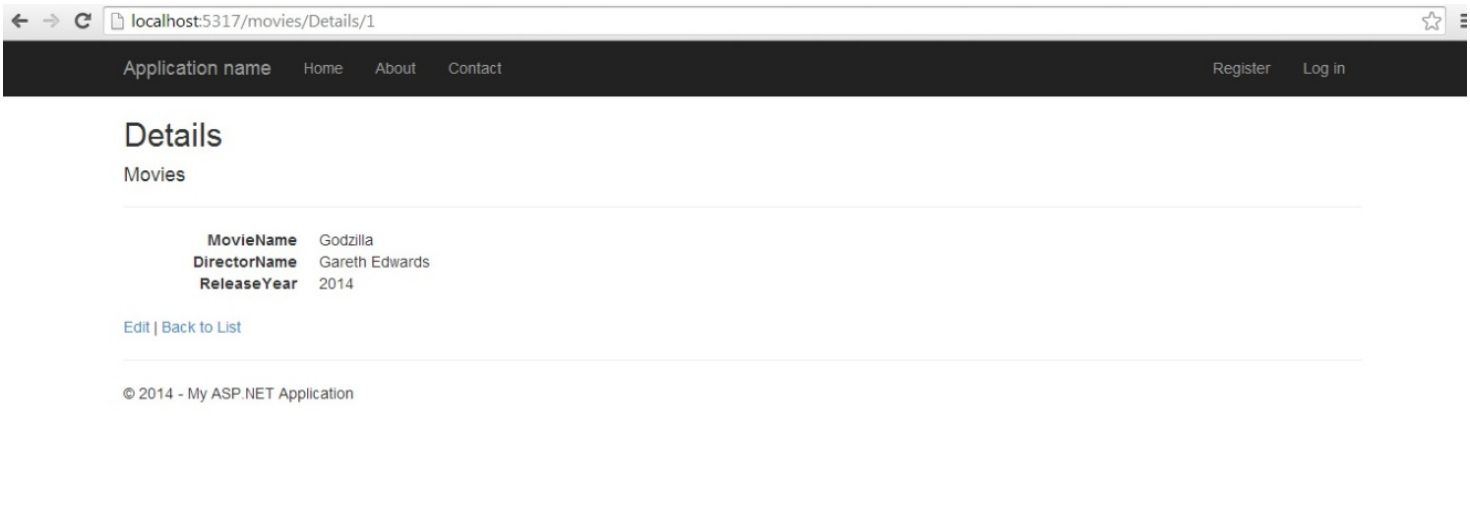


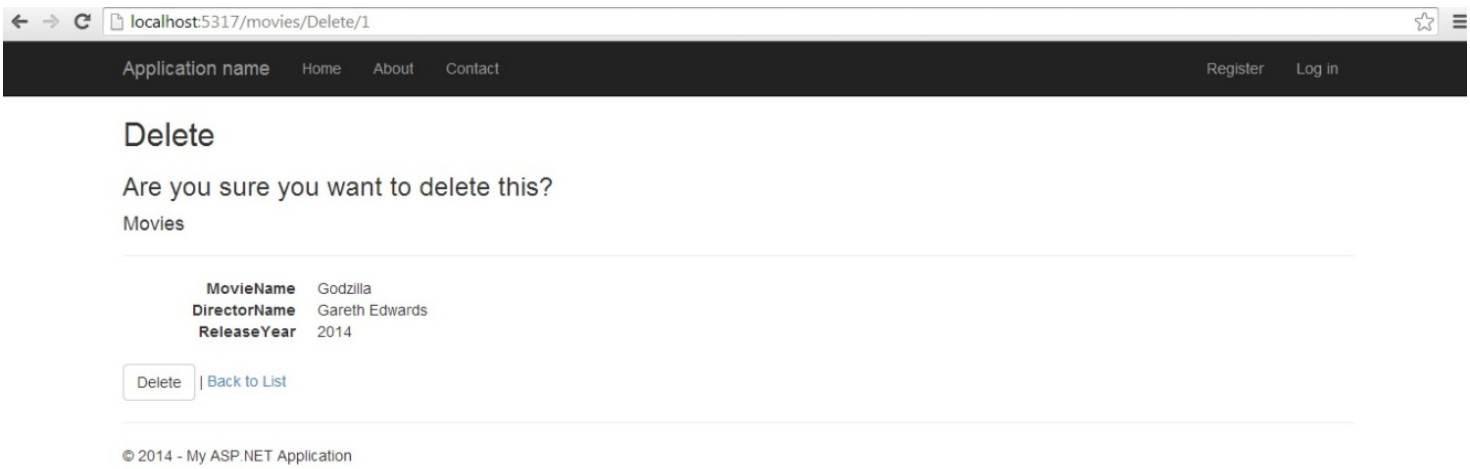


So, as you can see that my scaffolding has taken care of not only my new context, rather my all CRUD operation which you can verify as shown below. So, below is the Edit page.

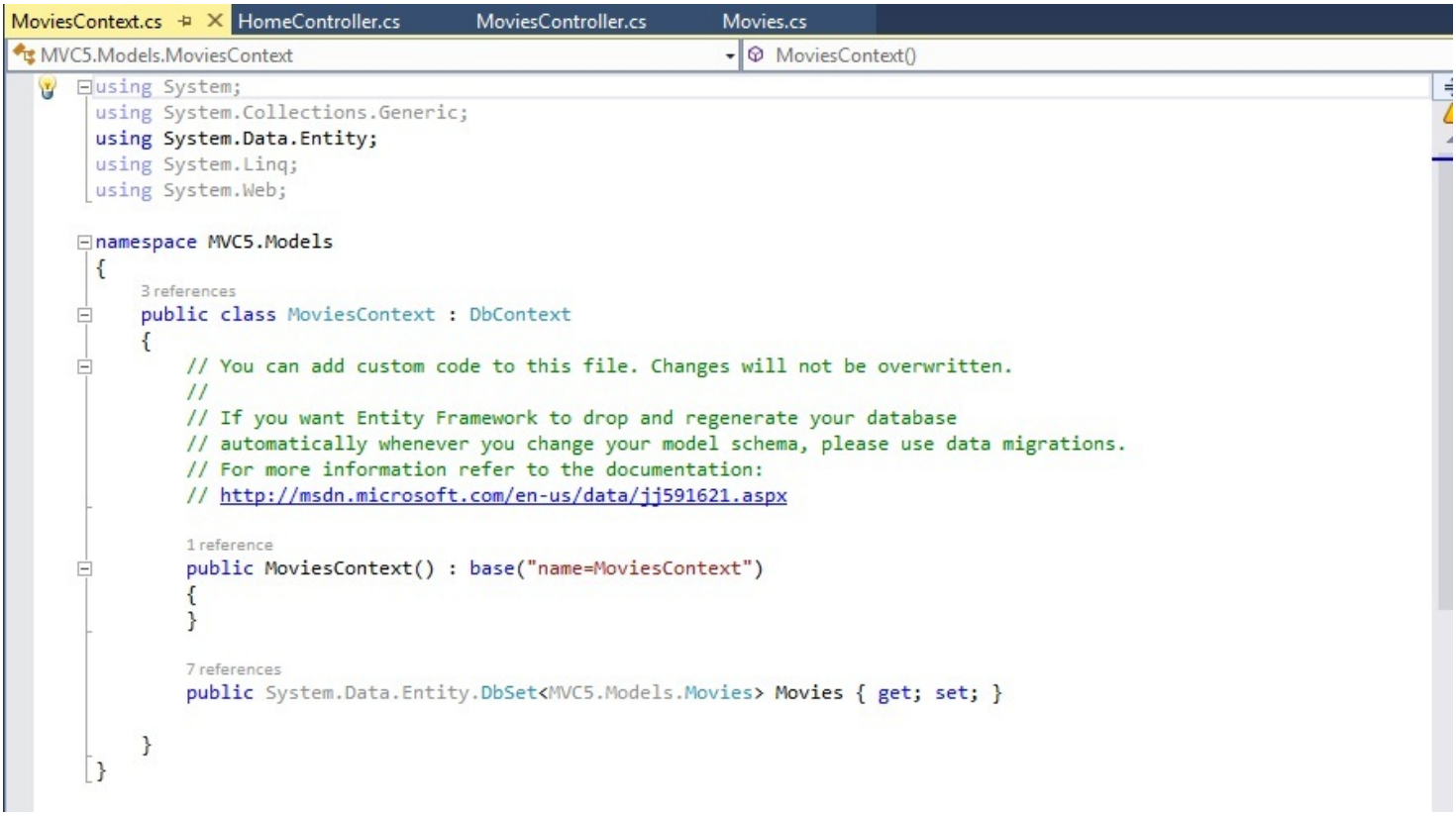


Similarly, we have details and delete page.





Now, if you go ahead and see the context file, then you can see below setting



Now, same entry is also added in the web.config file.

```
<connectionStrings>
  <add name="DefaultConnection" connectionString="Data Source=(LocalDb)\v11.0;AttachDbFilename=|DataDirectory|\aspnet-MVC5-20140619113346.mdf;Initial Catalog=aspnet-MVC5
    providerName="System.Data.SqlClient" />
  <add name="MoviesContext" connectionString="Data Source=(localdb)\v11.0; Initial Catalog=MoviesContext-20140621090938; Integrated Security=True; MultipleActiveResultSe
    providerName="System.Data.SqlClient" />
</connectionStrings>
```

Now, let's go ahead and check its new

Identity Feature.

ASP.Net Identity:-

So, while we were building our application, we have used membership provider for our application. Now, one point to note here, membership provider has come a long way since its inception. There have been many changes done to this in order to support today's web demand means these days web has become more social friendly and everywhere, you find this integration. So, considering these demands, web applications also need to provide similar experience so that users can

login with their social media credentials. Now, to get this done Modern Membership Provider is extended to support these new features. Some of the salient points for the same are listed below.

- **One ASP.Net Identity System**: - It can be used across all the ASP.NET Frameworks like WebForms, MVC, Web Pages, Web API, and SignalR etc.
- **Ease of plugging-in profile data**: - The user's profile schema information can be integrated with the web application.
- **Persistence Control**: - ASP.NET Identity system stores all user information in the database.

- **Social Login Provider**: - Social Log-In providers such as Facebook, Twitter, Google, Microsoft Account and others can be easily added to the web application.
- **Windows Azure Active Directory (WAAD):** Now, the best thing is that the Login-in information from WAAD can also be used for authenticating a web application.
- **OWIN Integration**: ASP.NET Identity is fully compliant with OWIN Framework. OWIN Authentication can be used for login. OWIN stands for "Open Web Interface for .NET". Now,

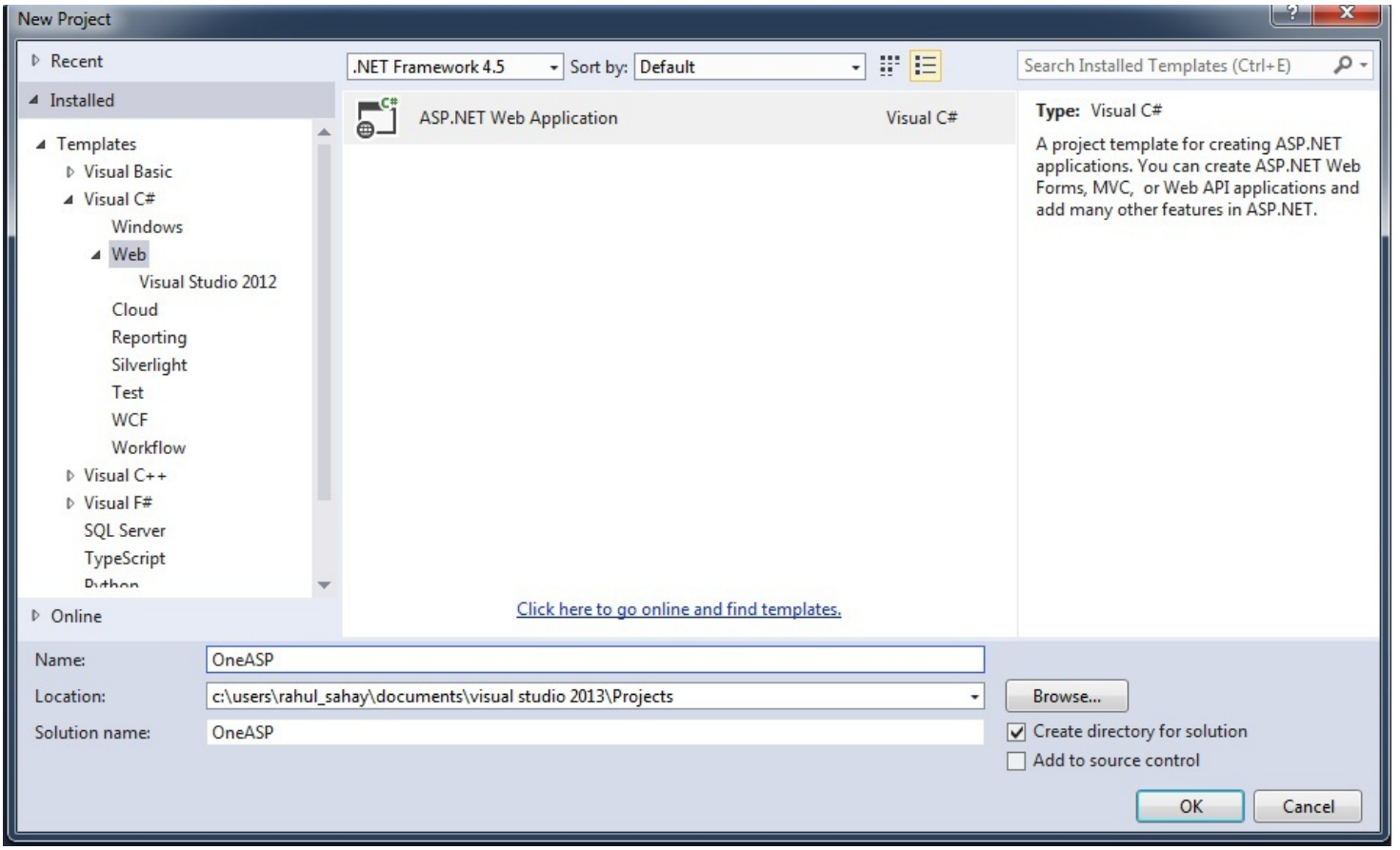
to do deep dive in OWIN framework, you can refer the following article ***http://owin.org/***

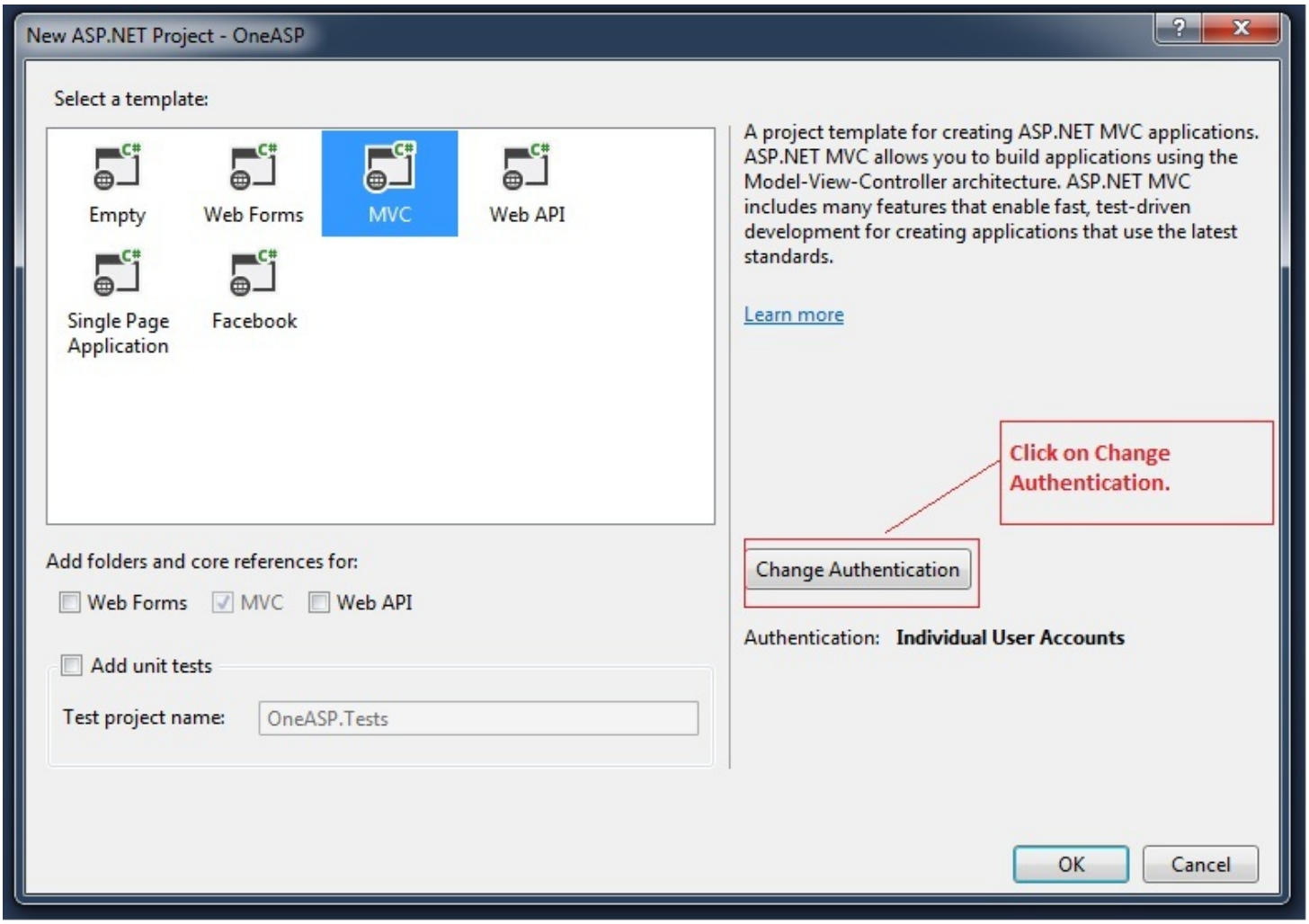
If you are using Visual Studio 2013, all these features are available in ASP.Net identity membership system by default; otherwise you can download the same from nuget package manager by searching with "***Microsoft.Aspnet.Identity.Core***" and "***Microsoft.Aspnet.Identity.EntityFram***

One ASP.Net:-

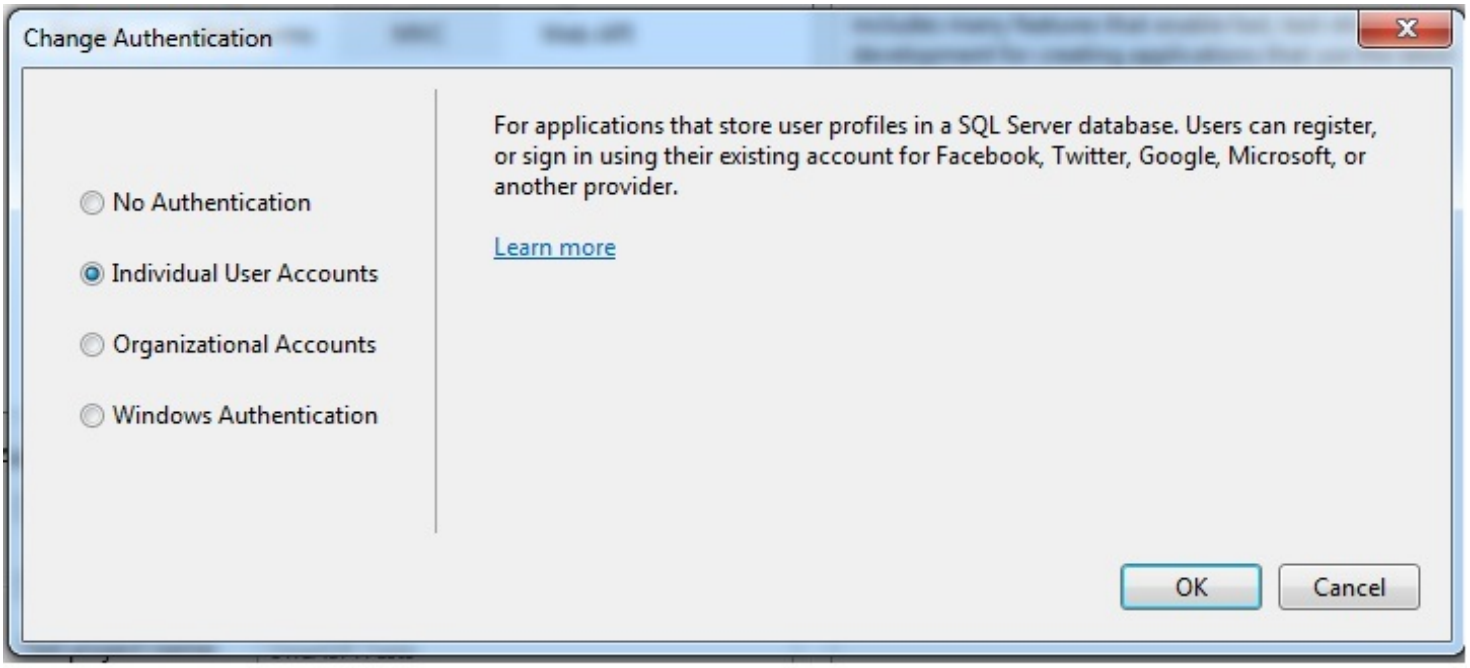
Now, One ASP.NET is a new integrated

project system for .NET Developers. This system makes it easier to work with multiple frameworks like Web Forms, MVC, Web API etc., in a single project. So basically using the One ASP.NET project system, you can use ASP.NET Web Forms and MVC together, and can easily add ASP.NET Web API and SignalR as well; in the same Web application. Now, let's go back to Visual Studio and start one small demo on the same. Let, me go ahead and create one new project with the name "***OneASP***" as shown below in the screen shot.


New Project
Recent
Installed
Templates
Visual Basic
Visual C#
Windows
Web
Visual Studio 2012
Cloud
Reporting
Silverlight
Test
WCF
Workflow
Visual C++
Visual F#
SQL Server
TypeScript
Online
.NET Framework 4.5
Sort by:
Default
Search Installed Templates (Ctrl+E)
ASP.NET Web Application
Visual C#
Type: Visual C#
A project template for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.
Click here to go online and find templates.
Name:
OneASP
Location:
c:\users\rahul_sahay\documents\visual studio 2013\Projects
Browse...
Solution name:
OneASP
Create directory for solution
Add to source control
OK
Cancel



Upon clicking on Change Authentication, it will produce several options as shown below in the screen shot



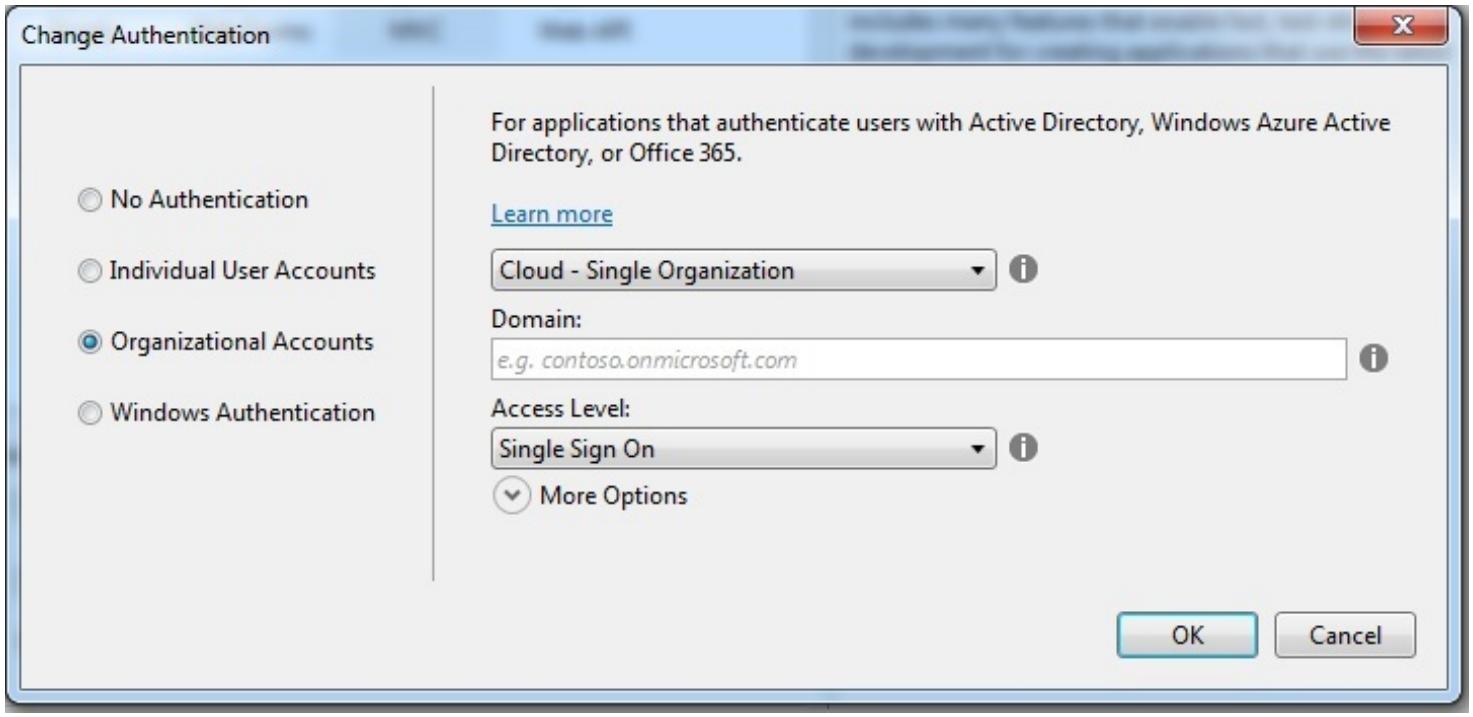
Now here you can see that there are four options provided to you.

No Authentication:- The application won't have any authentication.

Individual User Accounts: - Here, our Sql database will be used to store user's information. Now, one point to note here that, this option can be extended to support social profiles like Facebook,

Twitter etc.

Organizational Accounts: - Here, users can be authenticated either by using the user profiles stored in Active Directory, Windows Azure Active Directory, or Office 365. It also provides Single Sign-on access level to the application. The Organizational Accounts require the following details:



Now, when you click on both the dropdowns, it will present you with the following options

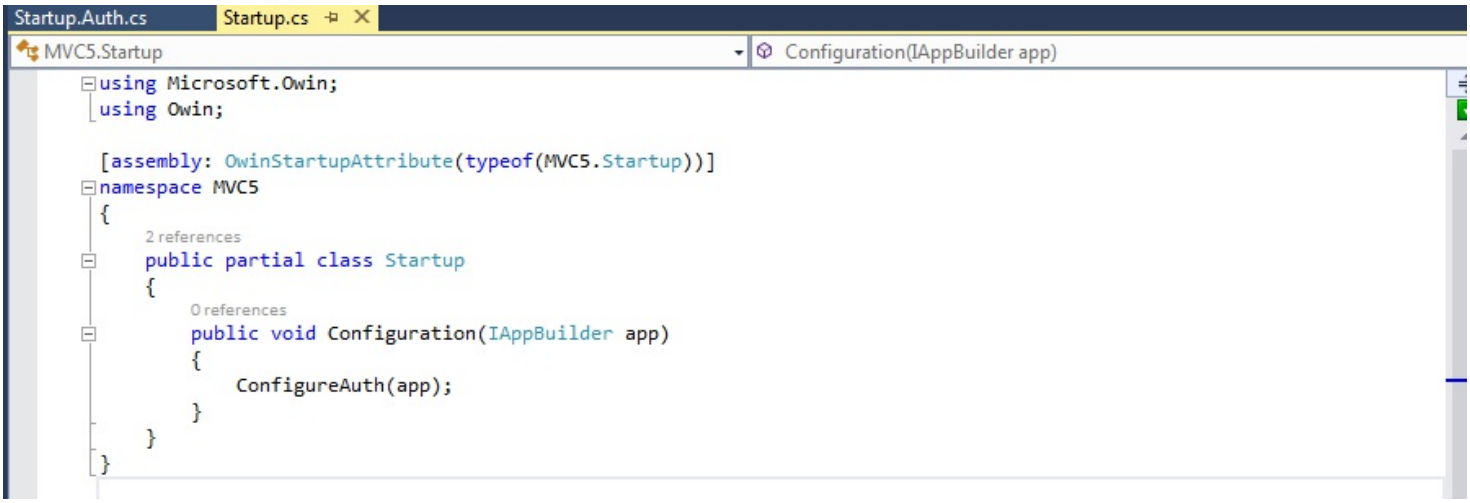
- **Cloud** :– Single Organization
- **Cloud-Multi Organization**: – Now, this option is applicable when you want to store the credentials information on the Windows Azure Activity Directory (WAAD.
- **On-Premises** - Used for the Active Directory on-premises.
- **Domain** - The WAAD domain for setting application in it.
- **Access Level** - The application needs to query or update directory information.

- **Application ID URI** – Basically, created by appending the project name to the domain.
- Now, after Organization Account structure the last option comes as
- **Windows Account**: - The basic one. Obviously, we have discussed the same while building the app in previous chapters. However, just for refreshing your theory, Windows Accounts are used for building Intranet Applications.

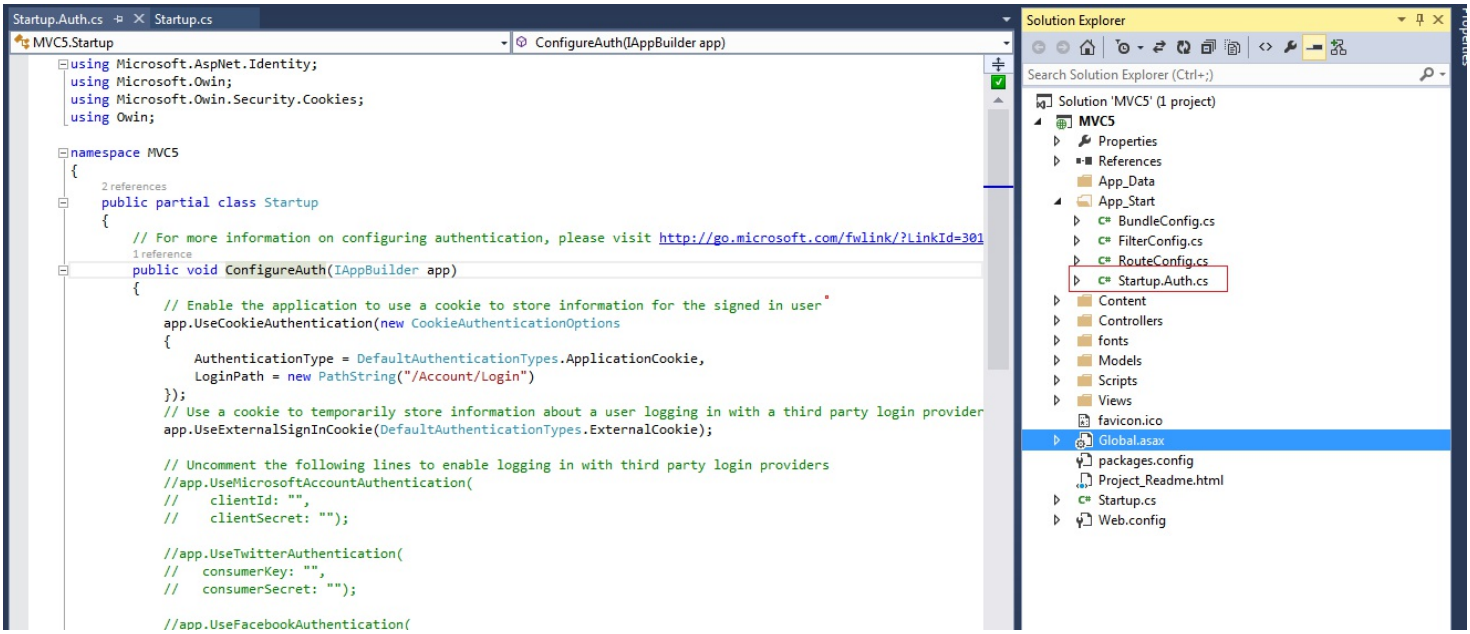
Also, apart from Organizational Accounts, there is **Windows Authentication** as well which supports

Intranet Applications. We have already discussed in detail about Windows Authentication.

Now, let's close this project, now go back to the project which we have created with the name "***MVC5***". Now, here this application is configured to use individual user accounts. So, let me go ahead and configure the application to use my social credentials as well for login. Now, here I'll be enabling it for google like I did earlier. But, here the process to do the same is bit different. So, here when you see the startup class, there is a call to "***configure auth***" as shown below in the screenshot.



Now, here when you do F12 on "***ConfigureAuth***" method, then it will take you to the class



Now, here I'll go ahead and uncomment the google API call as shown below and

then build the project.

```
    {
        // For more information on configuring authentication, please visit http://go.microsoft.com/fwlink/?LinkId=301
        1 reference
        public void ConfigureAuth(IAppBuilder app)
        {
            // Enable the application to use a cookie to store information for the signed in user
            app.UseCookieAuthentication(new CookieAuthenticationOptions
            {
                AuthenticationType = DefaultAuthenticationTypes.ApplicationCookie,
                LoginPath = new PathString("/Account/Login")
            });
            // Use a cookie to temporarily store information about a user logging in with a third party login provider
            app.UseExternalSignInCookie(DefaultAuthenticationTypes.ExternalCookie);

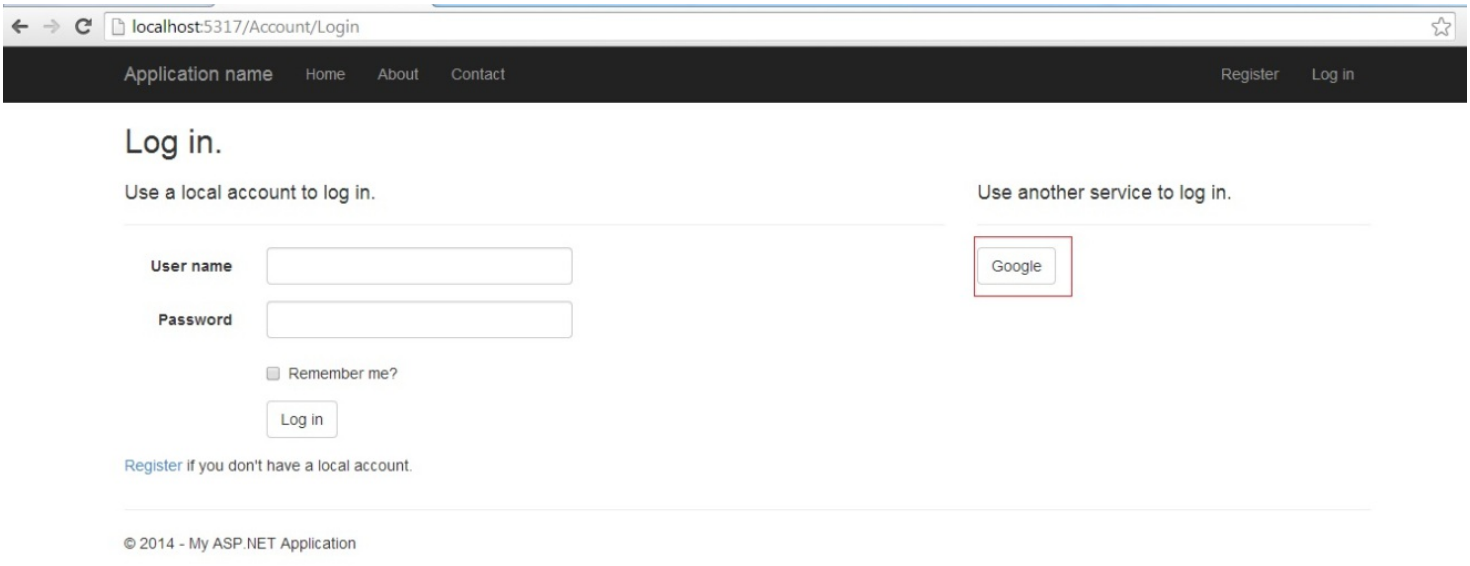
            // Uncomment the following lines to enable logging in with third party login providers
            //app.UseMicrosoftAccountAuthentication(
            //    clientId: "",
            //    clientSecret: "");

            //app.UseTwitterAuthentication(
            //   consumerKey: "",
            //   consumerSecret: "");

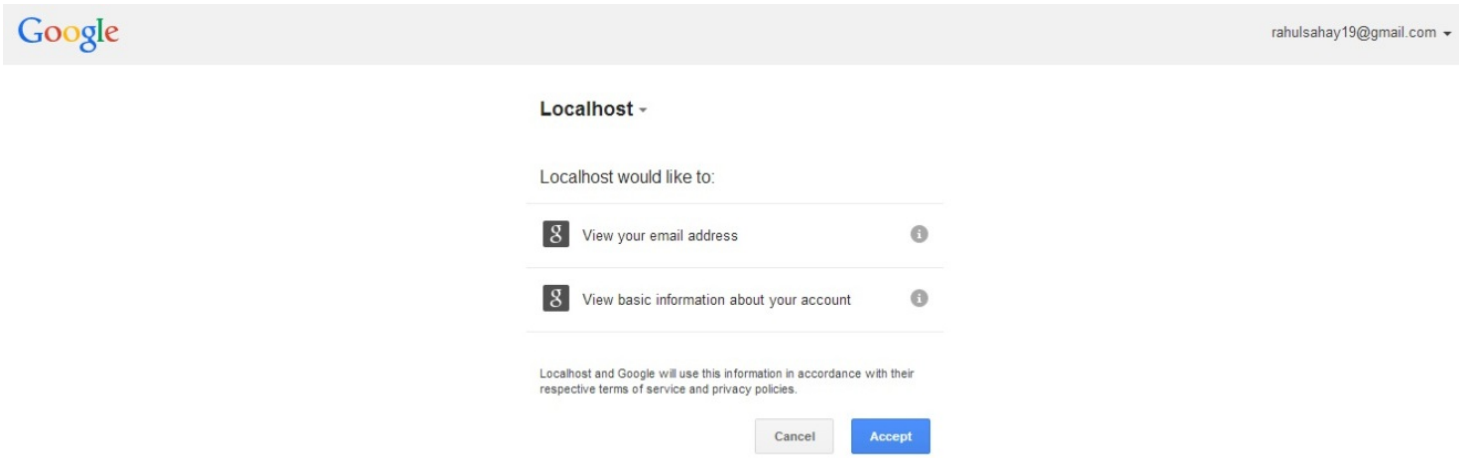
            //app.UseFacebookAuthentication(
            //   appId: "",
            //   appSecret: "");

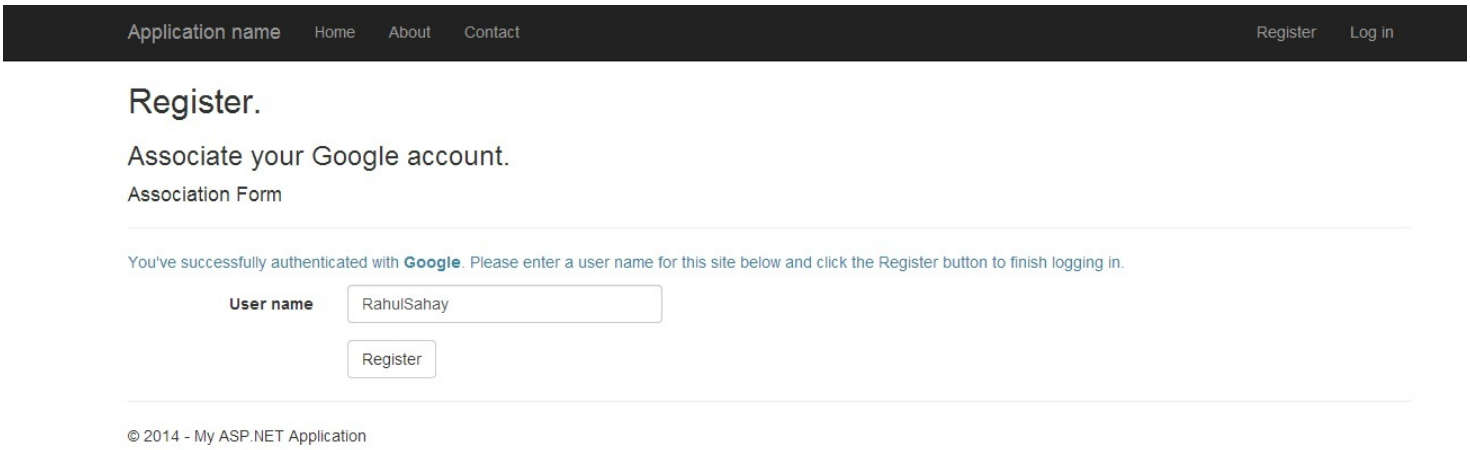
            app.UseGoogleAuthentication();
        }
    }
}
```

So, now when you run the app, you will see that google is also available to login into the application.

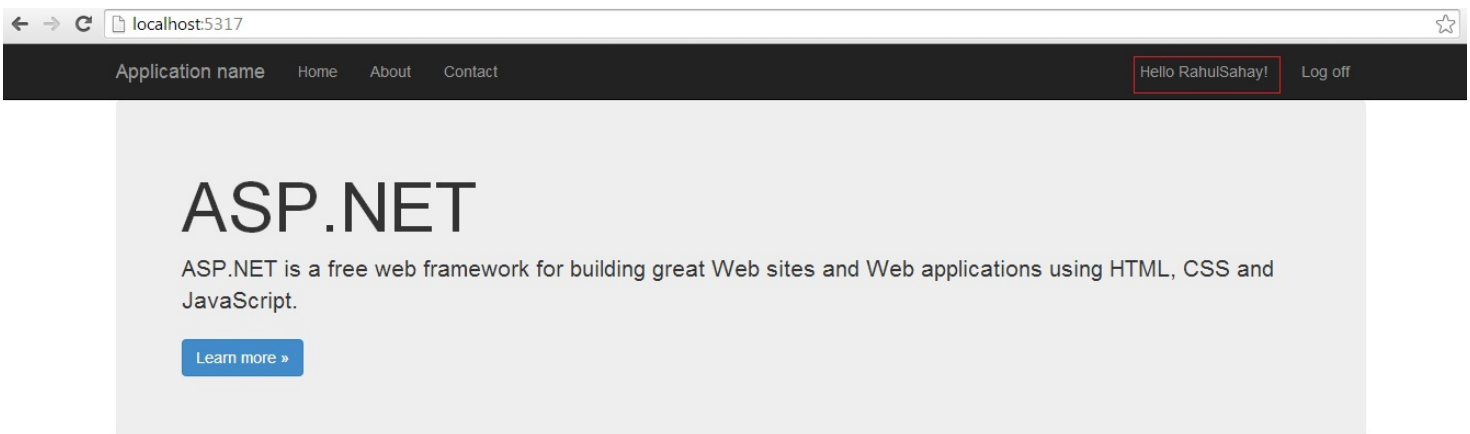


So, when you click on this link and provide your credentials, it will ask your permission for the 1st time as shown below and then jump back to application with your user id there





Now, when I click on Register, then I will land back to my application with my user Id as shown below.

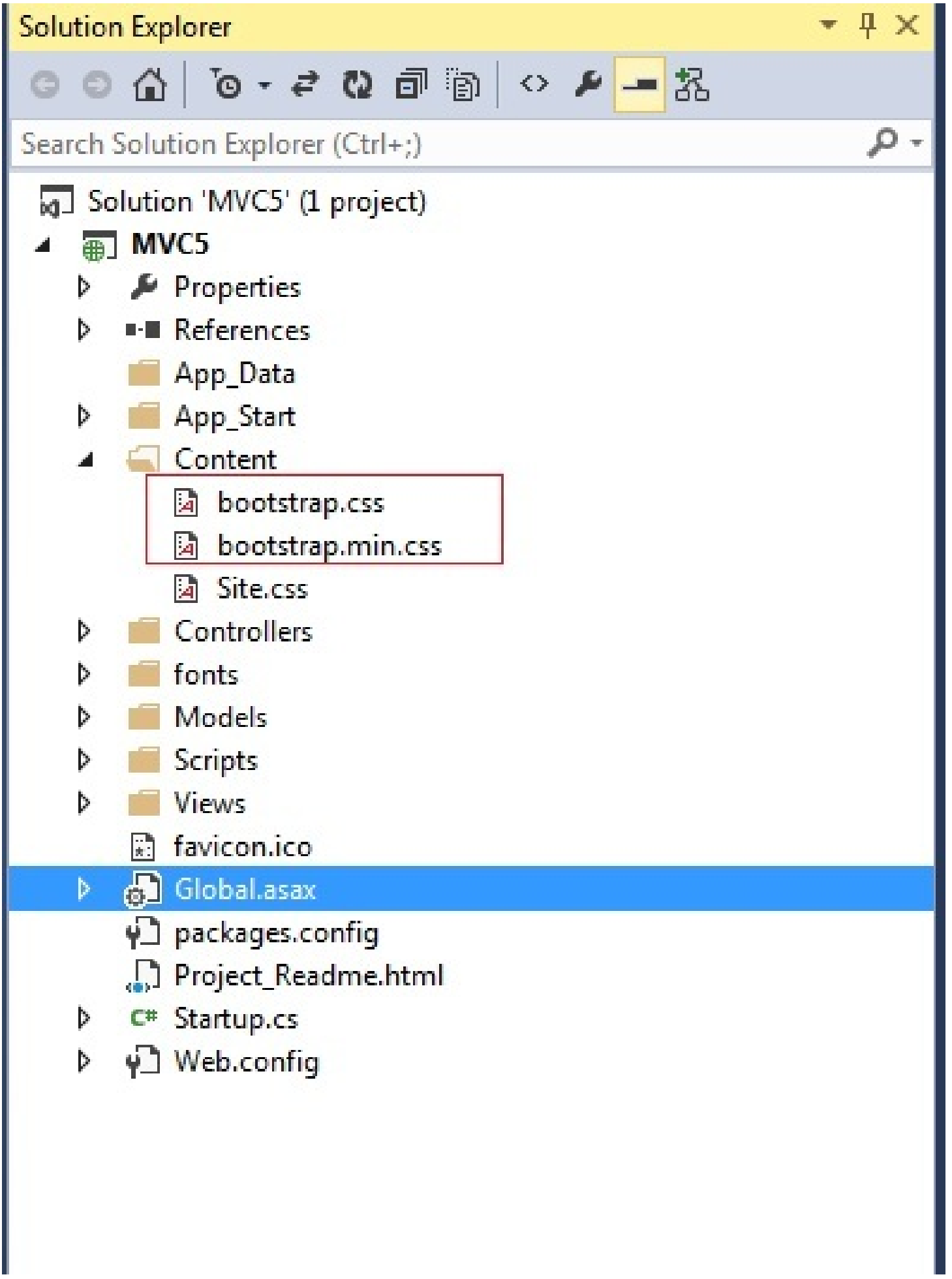


Similarly, you can enable other account settings, but for that you need to register your app with those social sites, so that

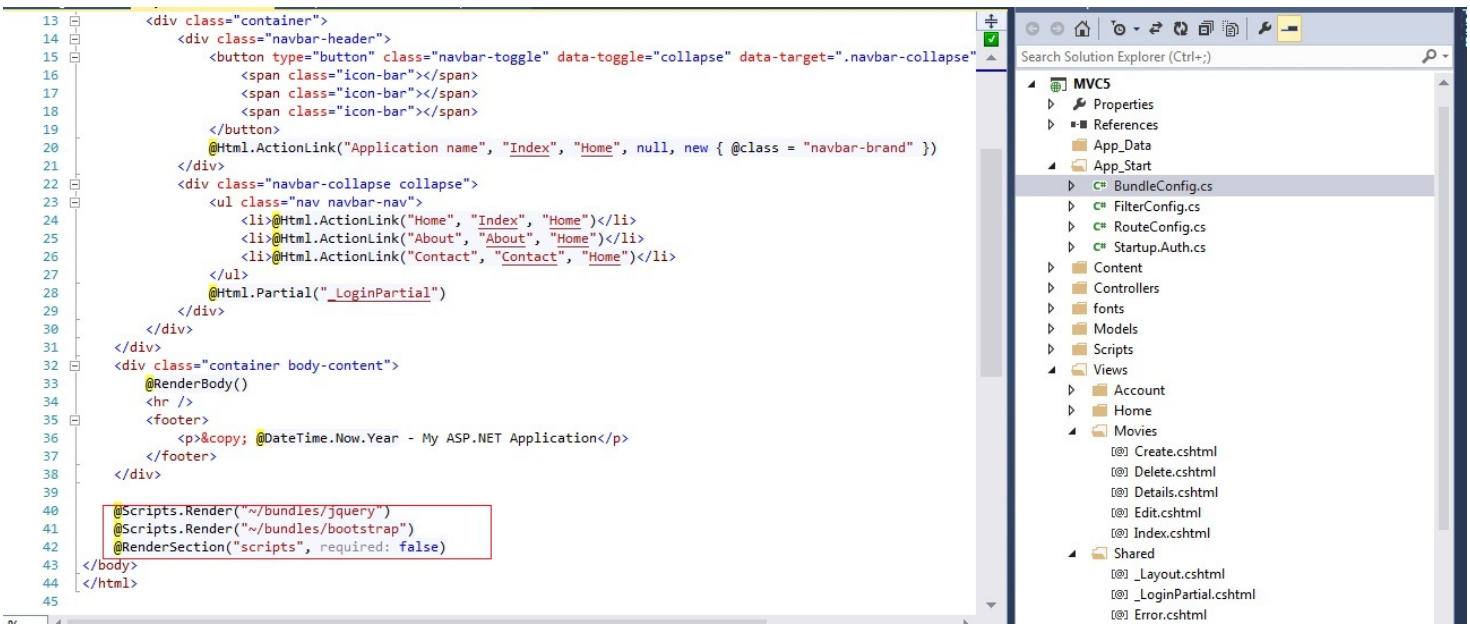
they can issue you one App id and it's key. Google is open as it doesn't require any sort of keys to enable open authentication.

Bootstrap:-

Now, in this case, MVC 5 application by default comes with bootstrap templates like shown below in the screen shot. I already explained this while designing my app in earlier chapters that bootstrap is a free collection of HTML and CSS based design templates. Also, bootstrap can be downloaded from **http://getbootstrap.com/**.

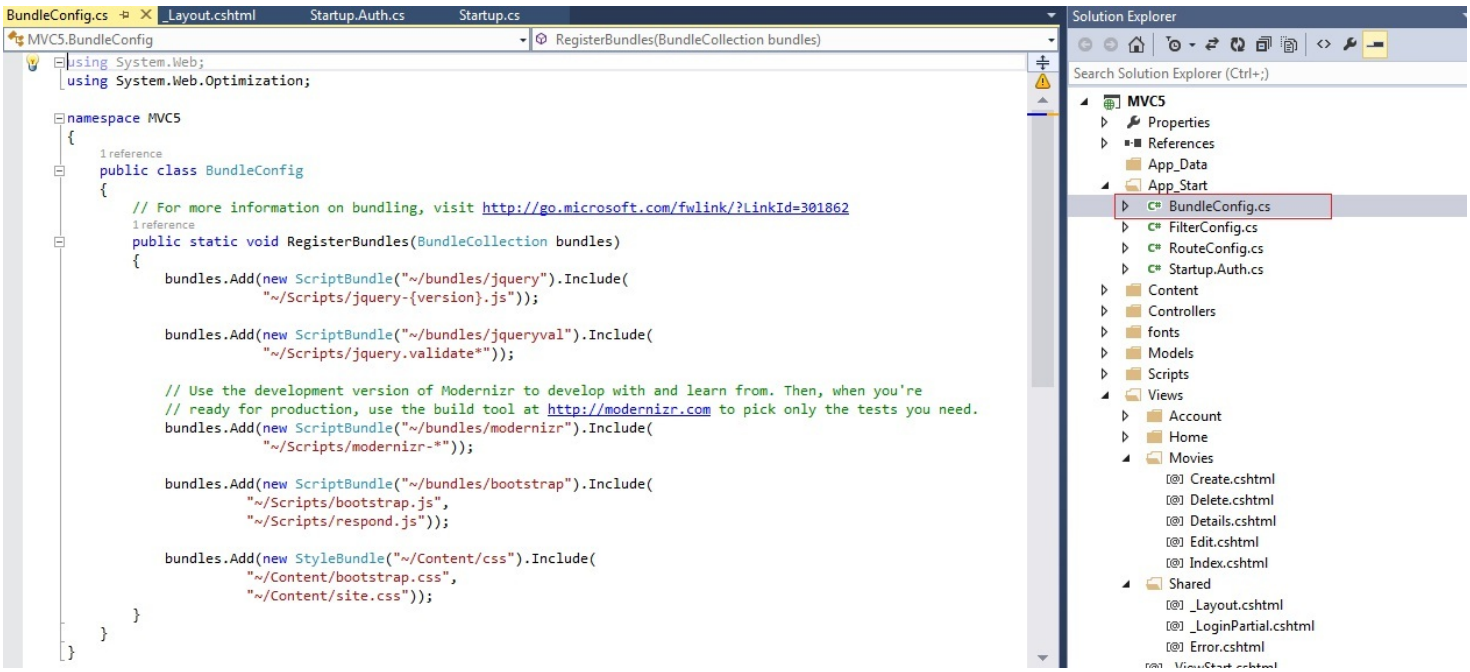

Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'MVC5' (1 project)
MVC5
Properties
References
App_Data
App_Start
Content
bootstrap.css
bootstrap.min.css
Site.css
Controllers
fonts
Models
Scripts
Views
favicon.ico
Global.asax
packages.config
Project_Readme.html
Startup.cs
Web.config

Now, if you open the layout page of the application, you will see these scripts have already included. Obviously; you can go ahead and use many other bootstrap widgets to give life to your application.

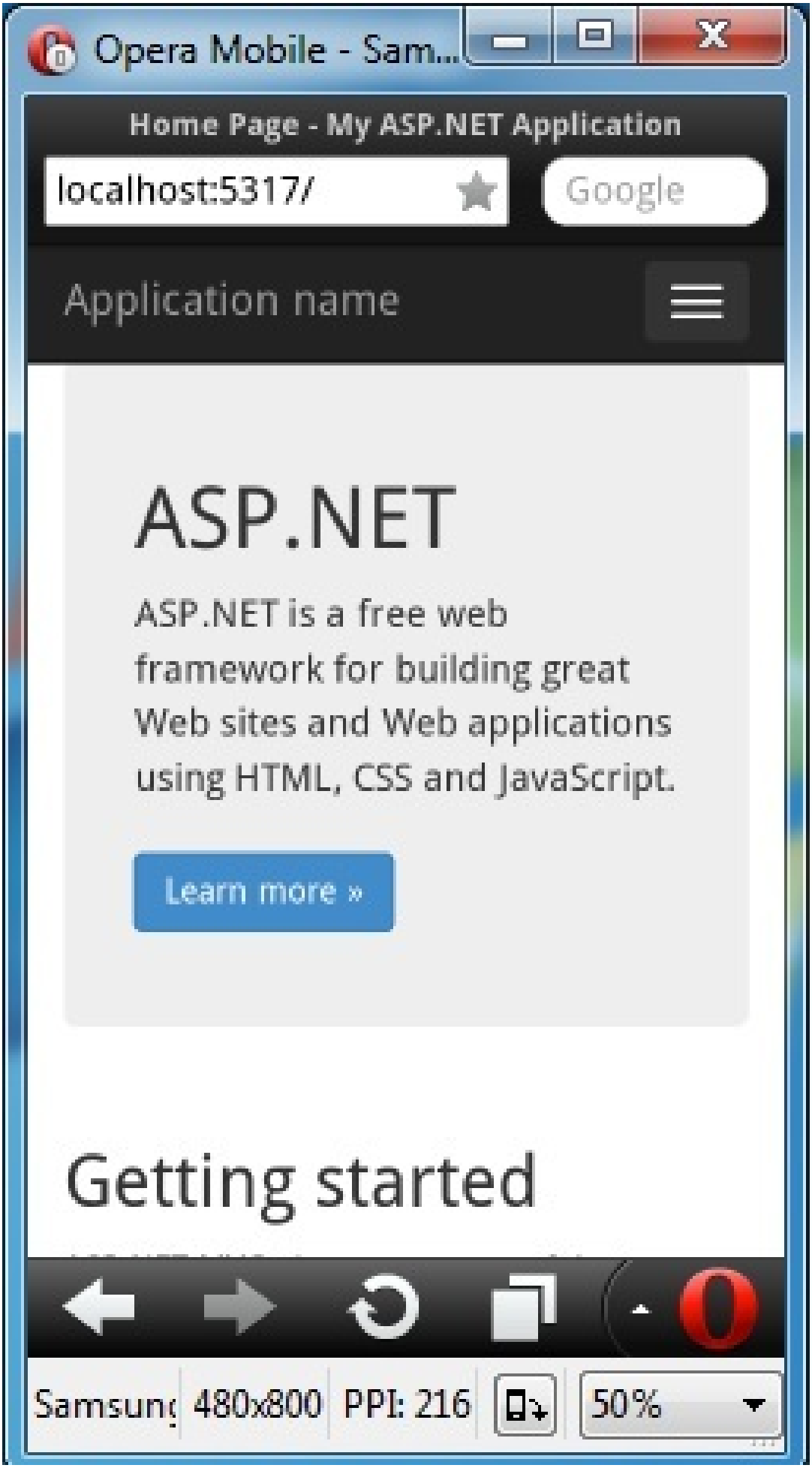


Now, as you can see in the above screen shot that bootstrap has been included with the bundle query. Now, to examine

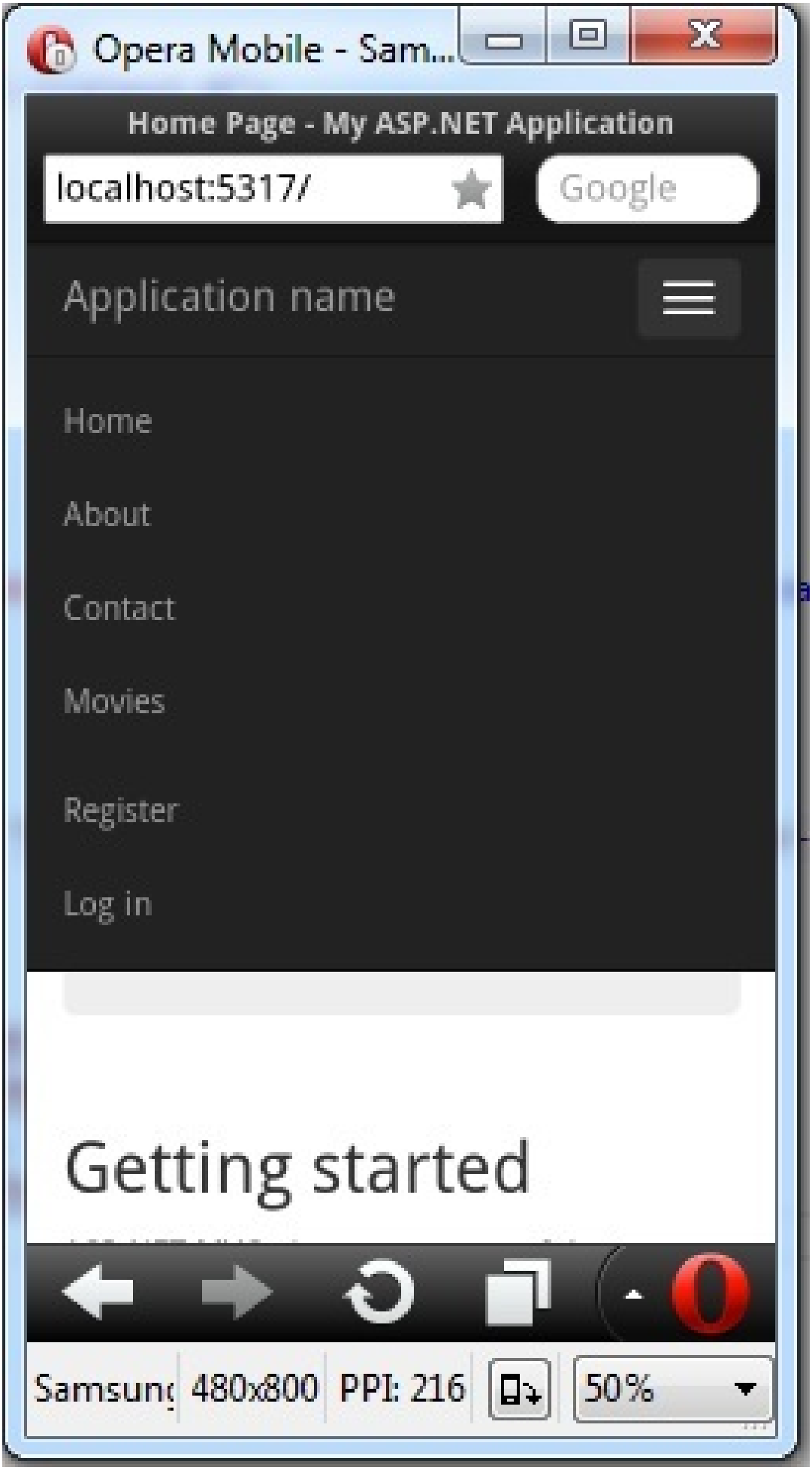
the same you can see the "***BundleConfig.cs***"



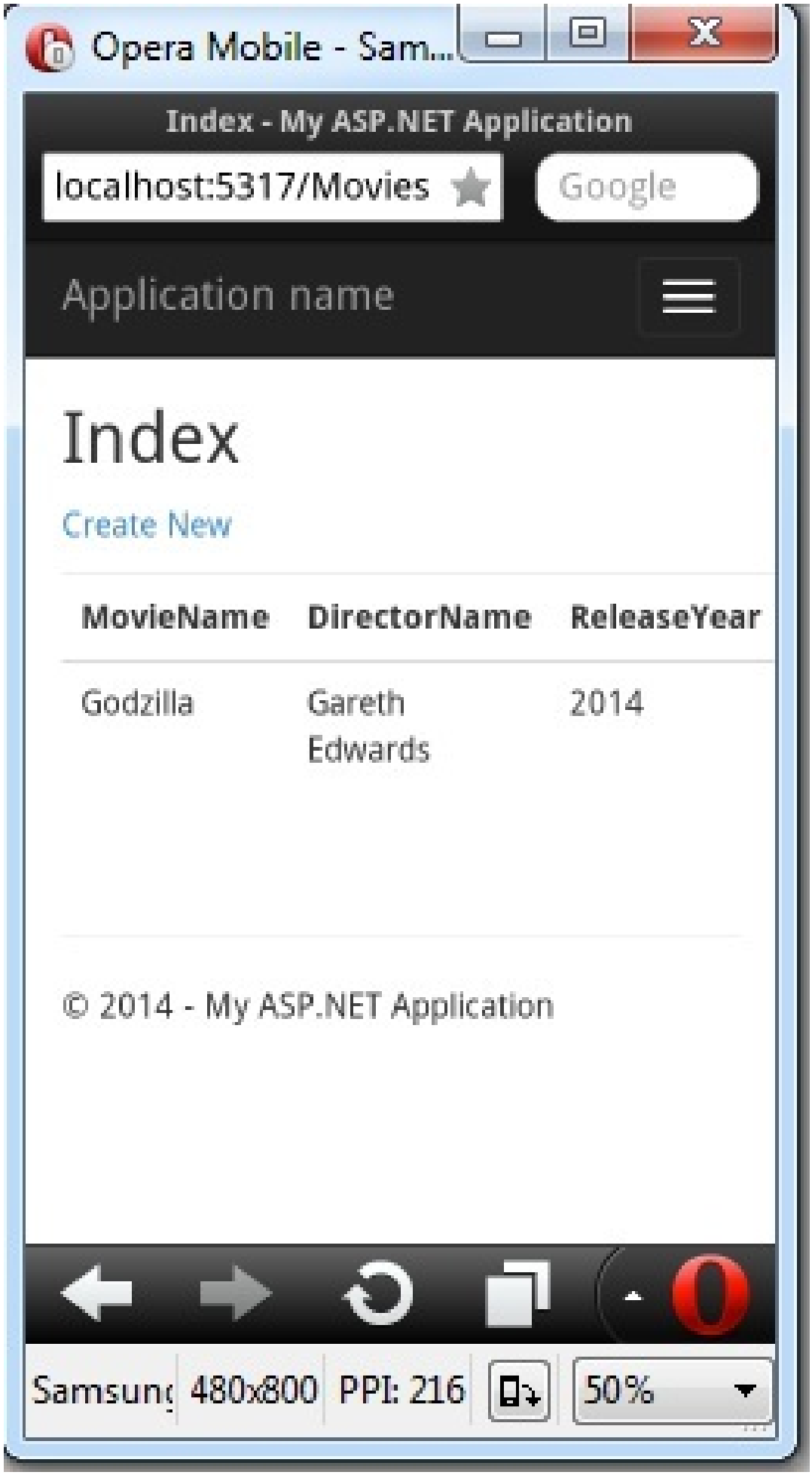
So, now one more point to note here is that MVC 5 by default supports mobile feature as shown below.


Opera Mobile - Sam...
Home Page - My ASP.NET Application
localhost:5317/
Google
Application name
ASP.NET
ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.
Learn more »
Getting started
Samsun
480x800
PPI: 216
50%

You might have noticed that all menus have been stacked at the top right corner of the screen. Now, when you click on this link, it will list all the menu items as a drop down.


Opera Mobile - Sam...
Home Page - My ASP.NET Application
localhost:5317/
Google
Application name
Home
About
Contact
Movies
Register
Log in
Getting started
Samsung
480x800
PPI: 216
50%

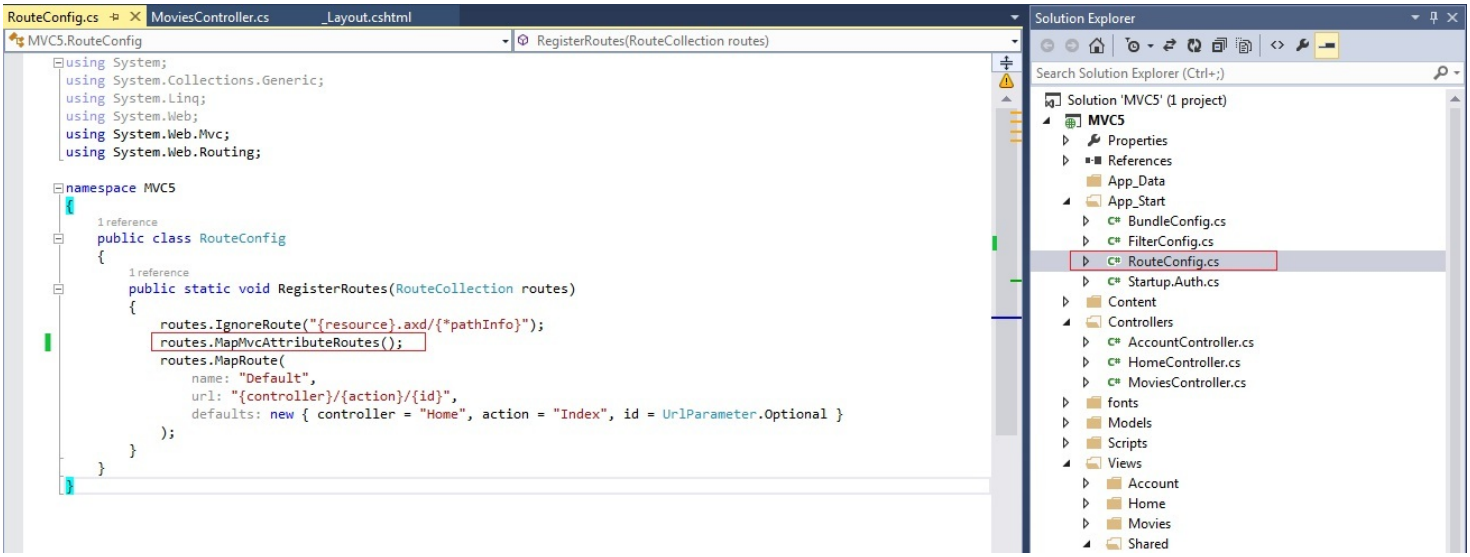
When you click on the "***Movies***" link here then it will produce the page which we have made with the new scaffolding template as shown below.


Opera Mobile - Sam...
Index - My ASP.NET Application
localhost:5317/Movies
Google
Application name
Index
Create New
MovieName
DirectorName
ReleaseYear
Godzilla
Gareth Edwards
2014
© 2014 - My ASP.NET Application
Samsung
480x800
PPI: 216
50%

Now, obviously there is always a room for perfection by overriding the current behavior. So, I leave this UX part on readers; how they want UI to look like.

Attribute Routing:-

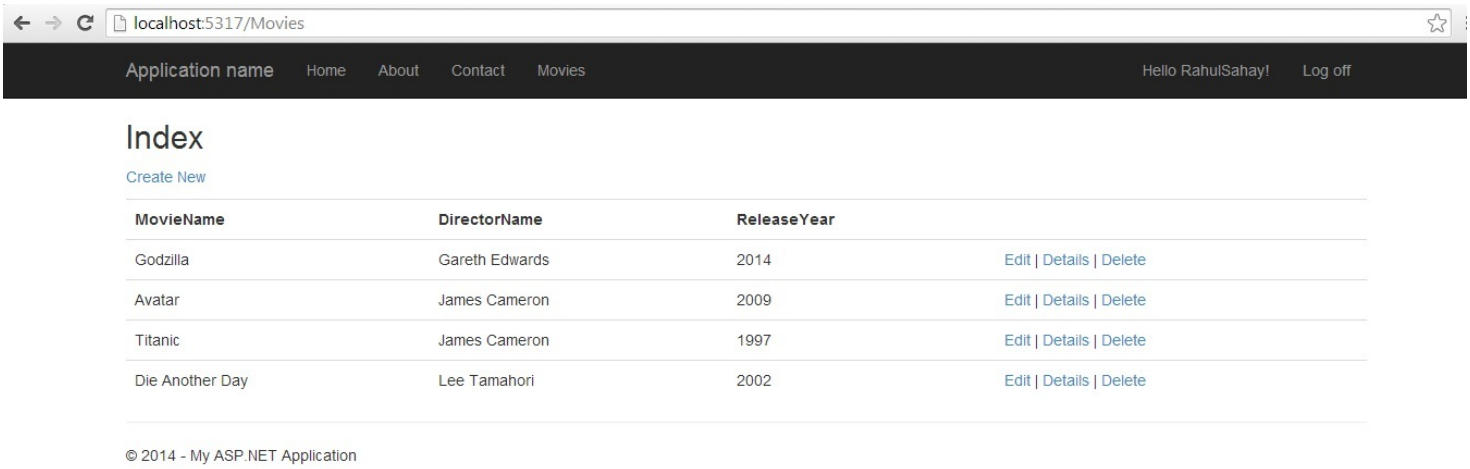
In MVC 5, we have gained more control over URI routing. We can implement the route definition along with the action methods in the controller class, using "**attributes**". In other words, we use attributes to define routes. To enable attribute based routing, the "**RegisterRoutes**" method needs to be changed:



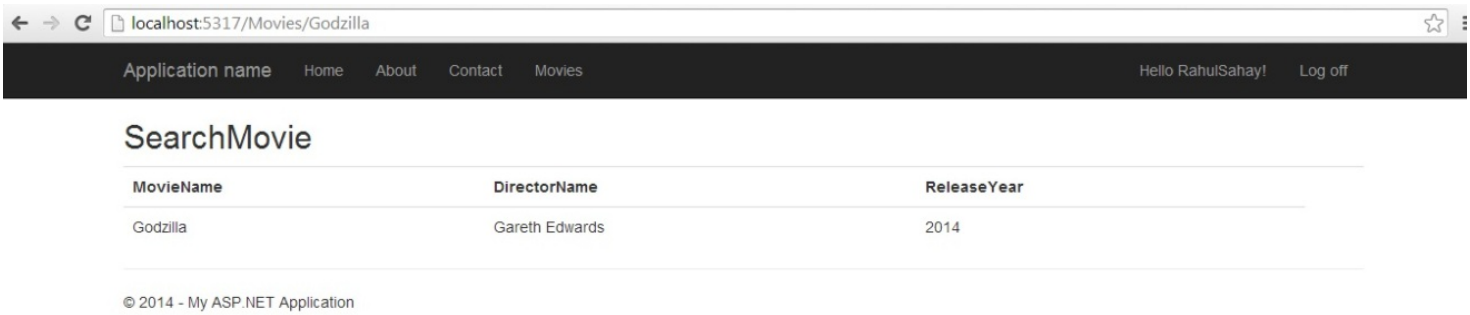


Now, after this setting is done, I need to write one method, let's say in the movies controller itself to filter the results. Since, I have added few movies here, so can filter the same based on movie name. So, for that I'll write below method

```
[Route("Movies/{MovieName}")]
        public ActionResult SearchMovie(string MovieName)
        {
            var model = db.Movies.Where(m => m.MovieName.StartsWith(MovieName));
            return View(model);
        }
```



Then, I'll navigate the ***http://localhost:5317/Movies/Godzilla***, then it will filter the results based on this parameter and produce the below output.

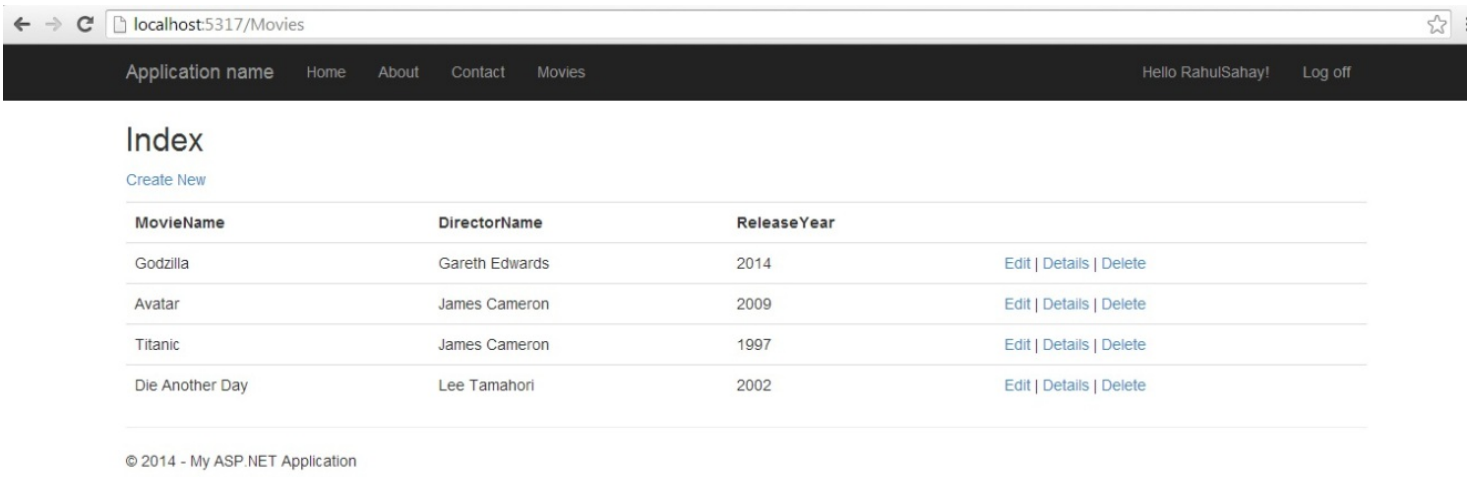


Now, here we can also define default value in the Route Parameters. Now, this is really simple, you just need to supply

the parameter value in the method as shown below.

```
[Route("Movies/SearchMovie/{MovieName=Godzilla}")]
        public ActionResult SearchMovie(string MovieName)
        {
            var model = db.Movies.Where(m => m.MovieName.StartsWith(MovieName));
            return View(model);
        }
```

So, now when I navigate the "***http://localhost:5317/Movies***" url, then it will display the complete list



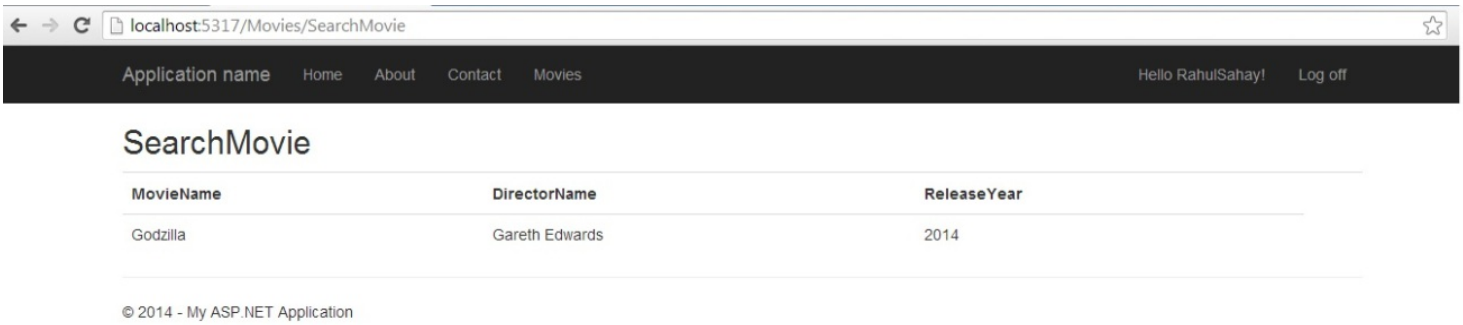
But, when I navigate "***http://localhost:5317/Movies/SearchM***

then it will produce me the below output



And if simply navigate "***http://localhost:5317/Movies/SearchM***

then it will produce me the default value as shown below



So, with route attributes in place, it gives much flexibility to developers, to design the application more route or URI

oriented.

Filter Overrides:-

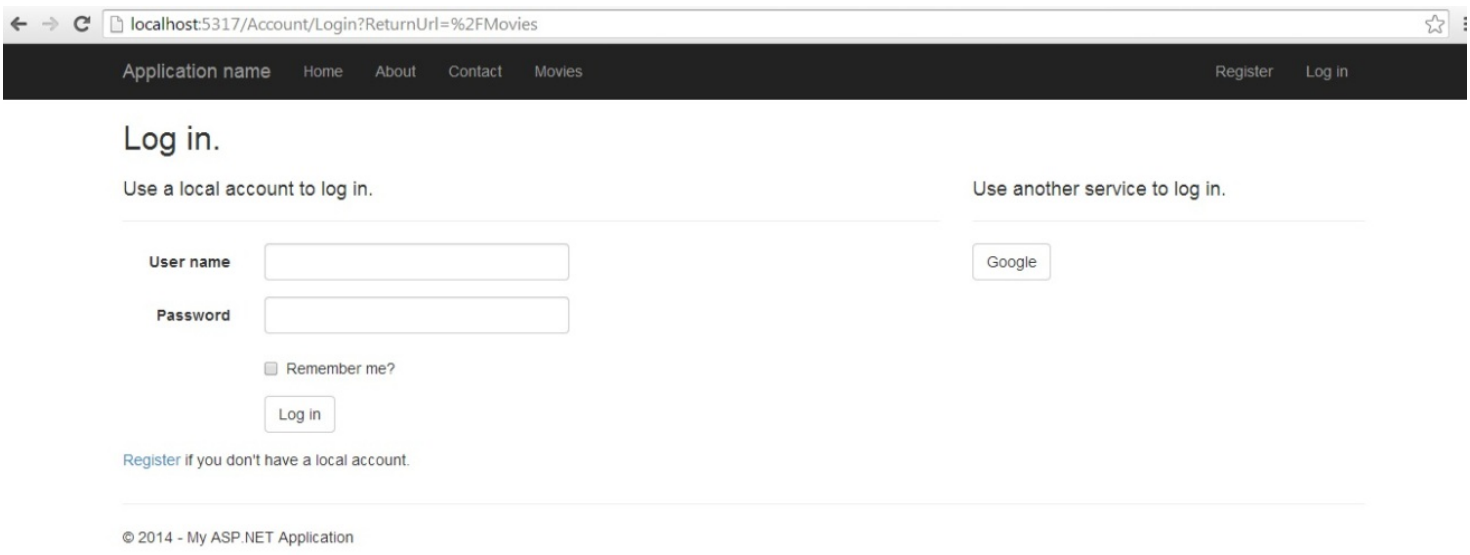
Now, filter overrides are just an extension to filters. In order to understand this concept better, let's suppose you want to authenticate users for some actions inside the controller or consider the entire controller actions you want to authenticate. Now, in this case you will apply "***[Authorize]***" attribute in front of the controller as shown below.

```
namespace MVC5.Controllers
{
    [Authorize]
    0 references
    public class MoviesController : Controller
    {
        private MoviesContext db = new MoviesContext();

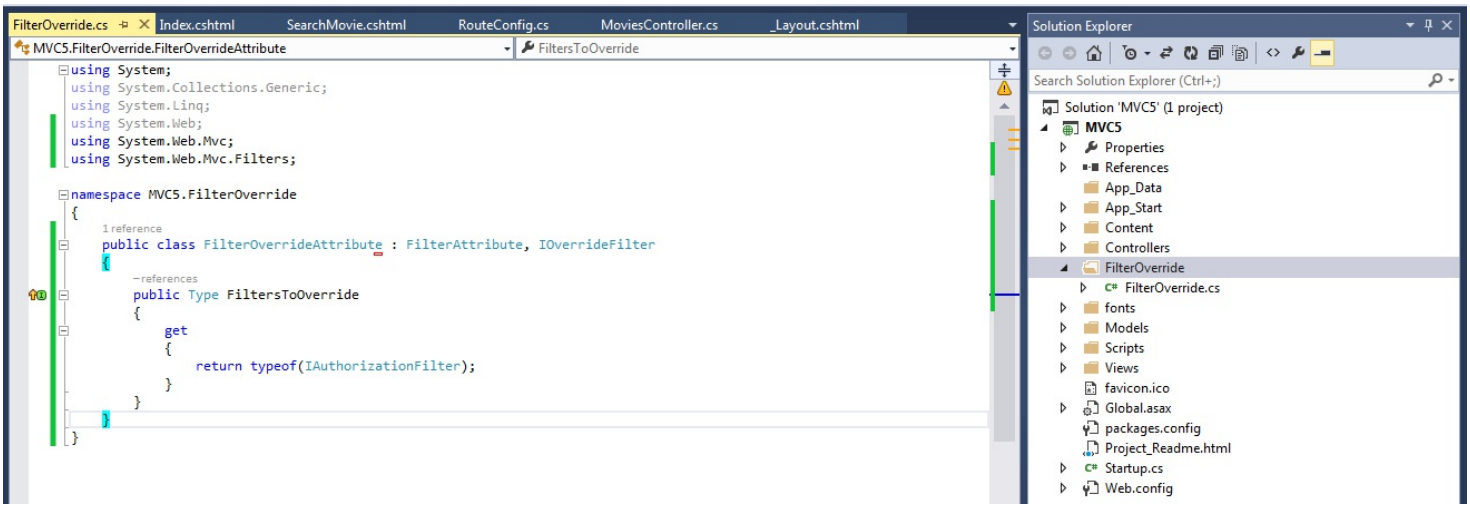
        // GET: /Movies/
        0 references
        public ActionResult Index()
        {
            return View(db.Movies.ToList());
        }

        [Route("Movies/SearchMovie/{MovieName=Godzilla}")]
        0 references
        public ActionResult SearchMovie(string MovieName)
        {
            var model = db.Movies.Where(m => m.MovieName.StartsWith(MovieName));
            return View(model);
        }
```

Now, when you build the app and try to navigate the Movies link, then it will take you to the login page as shown below.



Now, consider there is a situation, where in you want to bar all actions but, you want to show Index Action to the anonymous user. Then in that case you need to implement Filter Override mechanism to achieve the same. So, for doing the same you need to implement one class file with the name "***FilterOverride***"



Now, the class "***FilterOverriddeAttribute***" is inherited

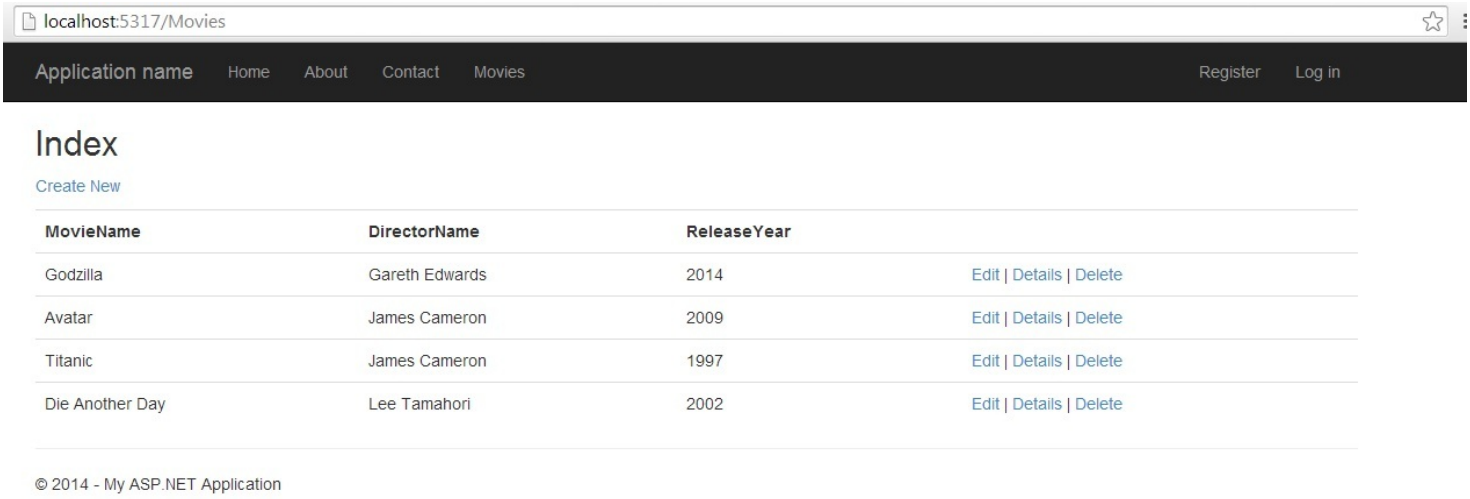
from the "***FilterAttribute***" class and implements "***IOverrideFilter***". Now, this interface is used to define the filters applied on the controller. The property "***FiltersToOverride***" returns the "***IAuthorizationFilter***" type.

Now, in front of that action method which you want to override just place the same, it will override the action.

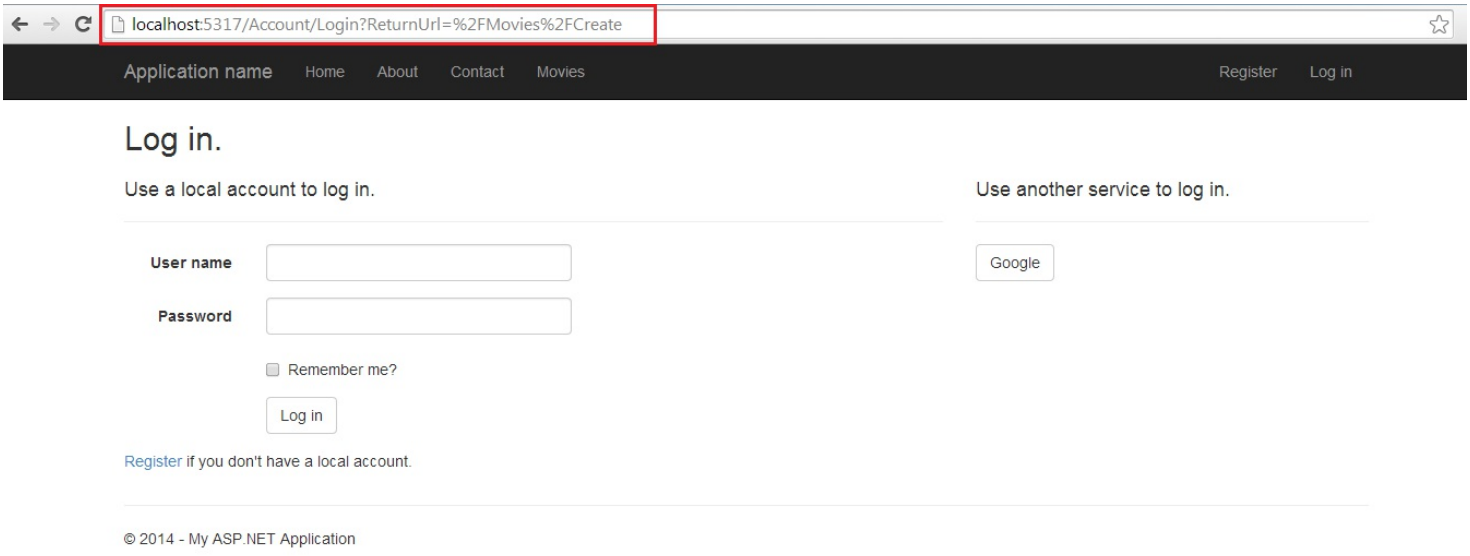
```
// GET: /Movies/
[FilterOverride]
0 references
public ActionResult Index()
{
    return View(db.Movies.ToList());
}
```

Now, when I build and refresh the page,

it will show me index page upon clicking on movies link, but other actions will need user authentication.



At the same time, when I click on "***Create New***" link, it will ask me to authenticate 1st as shown below.



Summary:-

So, as you have seen that MVC 5 is the extension of MVC 4. All these features we have already implemented while building our app. However, in MVC 5, things have become simpler and easier. It offers new thing like One ASP.NET, then improved scaffolding for MVC and Web API both. It also offers default support for bootstrap templates, which

we have implemented in our app by downloading additionally. Then, it also offers extension to filter attributes, which is very helpful in those cases where you want skip any action in the same controller.

Chapter 15: ASP.NET MVC 6

WHAT DO you find in this CHAPTER?

⇒ Introduction

⇒ Features

⇒ Prerequisites

⇒ Setting up VM

⇒ Getting Started with Visual Studio 14

⇒ Creating your 1st Visual Studio 14 MVC Project

⇒ Configurations in MVC Project

⇒ Project.JSON

⇒ Startup class

⇒ Adding a Class Library Project

⇒ Deployment to Azure

⇒ Bin

⇒ Self Hosting

⇒ Summary

Introduction:-

Hi Friends, so we have come to the end of this book wherein we'll discuss more about this new upcoming version of MVC. MVC 6 in many terms has given a big boost to the existing MVC. Let's discuss the same in detail. Now, in a nutshell "***MVC 6 = MVC + Web API + Web Pages***". So, MVC 6 has merged these three different frameworks, hence MVC 6 is a next version of MVC off

course, but it's also a next version of Web API 3 and Web Pages 4. So, basically speaking there will be a common framework which will let you do Model Binding, Routing, Validation, Dependency Injection and many more things. So, let's get started and discuss the same in detail.

Features:-

So what are you going to get with MVC 6?

- **One Set of concepts** – So, there will be a Framework which will do both Web UI and Web APIs. So, it will have

controllers which will return views and it can also return Web APIs. So, same controller can return formatted data JSON, xml etc.

- **Smooth Transition from Web Pages to MVC** – Let's consider a scenario that you have started building very simple web application with bunch of "cshtmls" files. Now, you can go ahead and put the code inline and scale it out. So, it will have all the MVC features like Areas, controllers and Unit Testing other relevant stuffs.
- **Built DI First**: - So, one of the main focuses of MVC 6 is DI

(Dependency Injection) First. Today in MVC we have dependency resolver and there is set of implementation for that to work with your favorite IOC (Inversion of Control). Then Web API has its dependency resolving mechanism. So, in MVC 6, there is centralized repository for DI which can be used by everybody. So, going forward it's not like that different DI Implementation mechanism for different IOCs. It will be Kind of DI Abstract frameworks which will give codebase to get system bootstrapped.

- **Based on new Request Pipeline in ASP.Net vNext**: - MVC 6 is built on new request pipeline which comes with “ASP.Net vNext”. It runs with a new Middleware pipeline which comes with ASP.Net vNext.
- **Run on IIS or Self Host**: -. Since MVC 6 run on a new pipeline, so it can be self-hosted as well; means it can be hosted outside of IIS. So, you don't need to have IIS in your box to host the application. Self Hosting also means it will have mini “CLR” version which is “KLR” in your hosting folder, which will actually create a runtime

environment for the application to run in any environment say MAC, or in Linux box etc.

- **Runs Cloud Optimized**: - Cloud optimized new version of .NET which is slimmed down and highly optimized. Consider your system which is running on 2.0 framework and you want to work on a new project which is latest and greatest but somehow you don't have option to download those frameworks and build that. So, with this option what you can do, you can download this particular framework in your project via nuget itself; which is going to

be mini version of CLR itself which you are needing not going to affect your machine settings at all. So, with this option you can actually run different projects targeting different frameworks side by side.

Now, apart from these features there are other major changes also done to MVC 6 .They are:-

- MVC 6 has no dependency on "***System.Web***". Hence, it's a leaner framework, with faster startup time and lower memory consumption.
- vNext apps uses a cloud-optimized

.NET framework. This means framework is about 11 megabytes in size compared to 200 megabytes for the full framework, and is composed of a collection of NuGet packages.

- The reduced size of the cloud-optimized framework makes it practical to deploy very smoothly.
- You can run apps side-by-side with different versions of the framework on the same machine.
- Your apps are fairly insulated from framework changes on the machine.
- You can make framework updates

for each app on its own.

- No errors when you deploy to production resulting from a mismatch between the frameworks.
- Providing the cloud-optimized framework in a collection of NuGet packages achieves the following goals:

1 You can include with an application only the parts of the framework you actually need. No unnecessary memory, disk space, loading time, etc.

2 vNext uses the "***Roslyn***" to compile code dynamically. We can edit a code file, refresh the browser, and see the changes

without rebuilding the project.

3 vNext is open source and cross platform which means it can run on MAC machines or Linux based machines as well.

4 The entire source code is already released as open source via the"***.NET Foundation***". You can also refer the source at "***https://github.com/aspnet***" and follow progress on vNext in real time. You can also send pull requests and contribute to the source code.

5 vNext is developed keeping cross platform in mind including an active collaboration with Xamarin to ensure that cloud-

optimized .NET applications can run on Mac or Linux on top of the Mono runtime.

However, I have also inserted one table below showing complete difference between normal vNext version and its cloud optimized version.

Feature	.Net vNext	.Net vNext (Cloud Optimized)
Cloud Ready	√	√
Modular Design	√	√
Dependency Injection	√	√
Side by Side		√
Open Source	√	√
Fast Start up		√
Low Memory		√
High Troughput		√
Fast Development	√	√
Consistent Tracing/Debugging	√	√

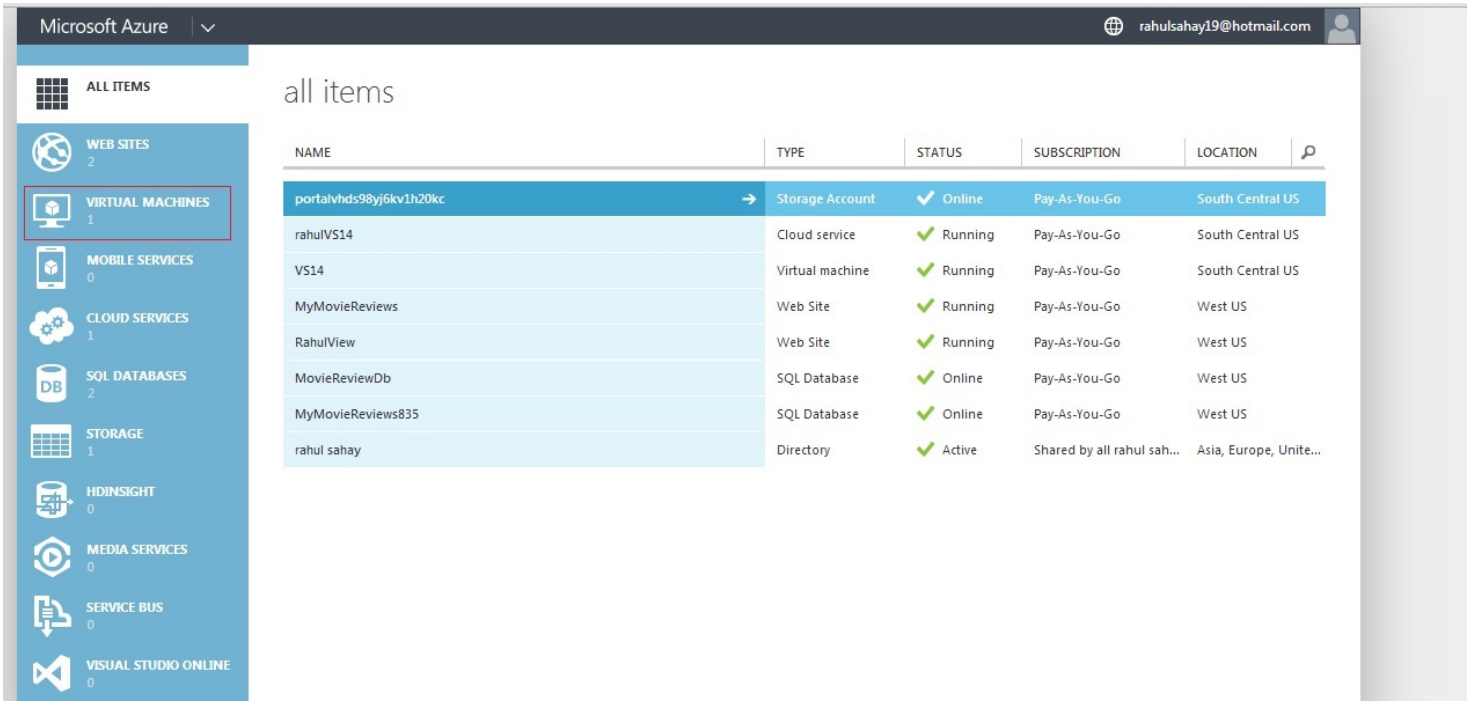
Prerequisites:-

It is recommended to install the Visual Studio "14" CTP on a VM (Virtual Machine) or fresh computer which can be formatted later on. There are known side-by-side compatibility issues with Visual Studio 2013. Now, to deploy the same in VM, you need to have Azure Subscription for that. Even if you don't have Azure subscription, you can enjoy these features for 1 month as a trial period. So, without wasting time, let's get started.

Setting up VM:-

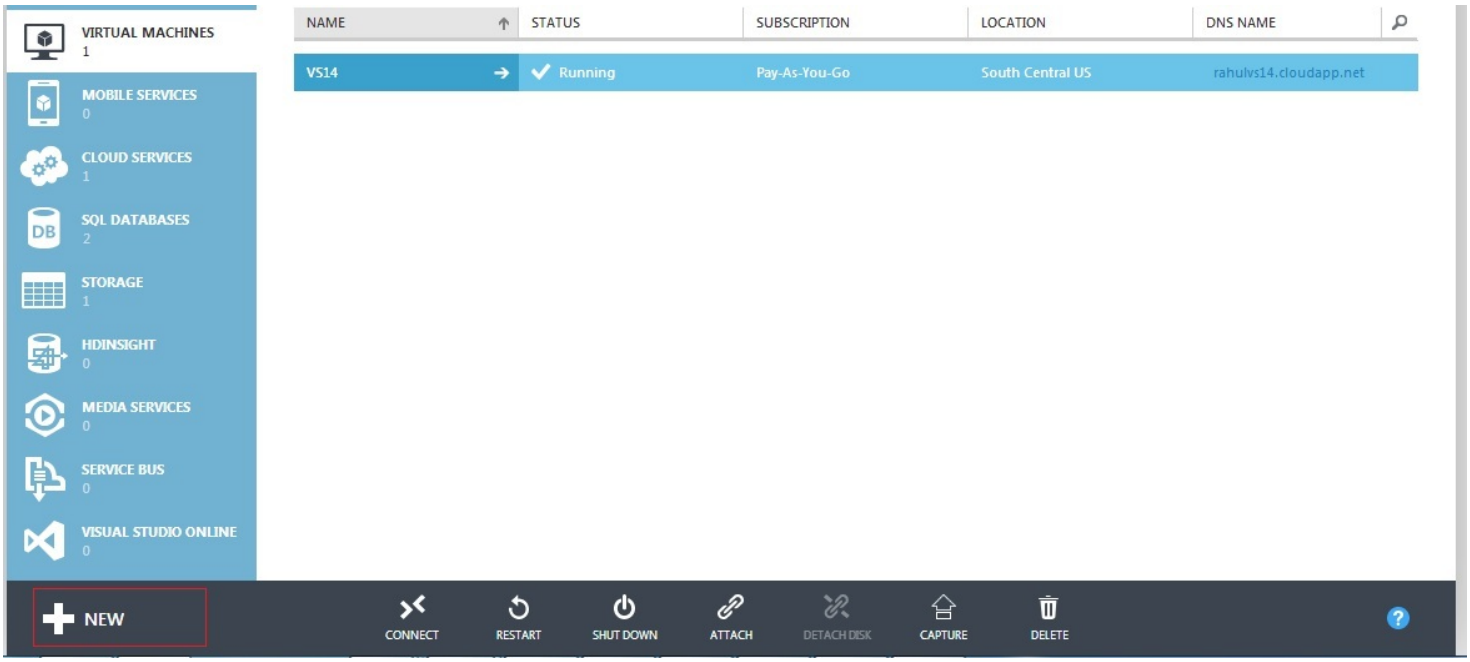
Since, I have an Azure subscription, so let me go ahead and create one Custom

VM for myself. Login into ***http://azure.microsoft.com/en-us/***, once you are logged in successfully, you will land on a dashboard page as shown below in the screen shot.

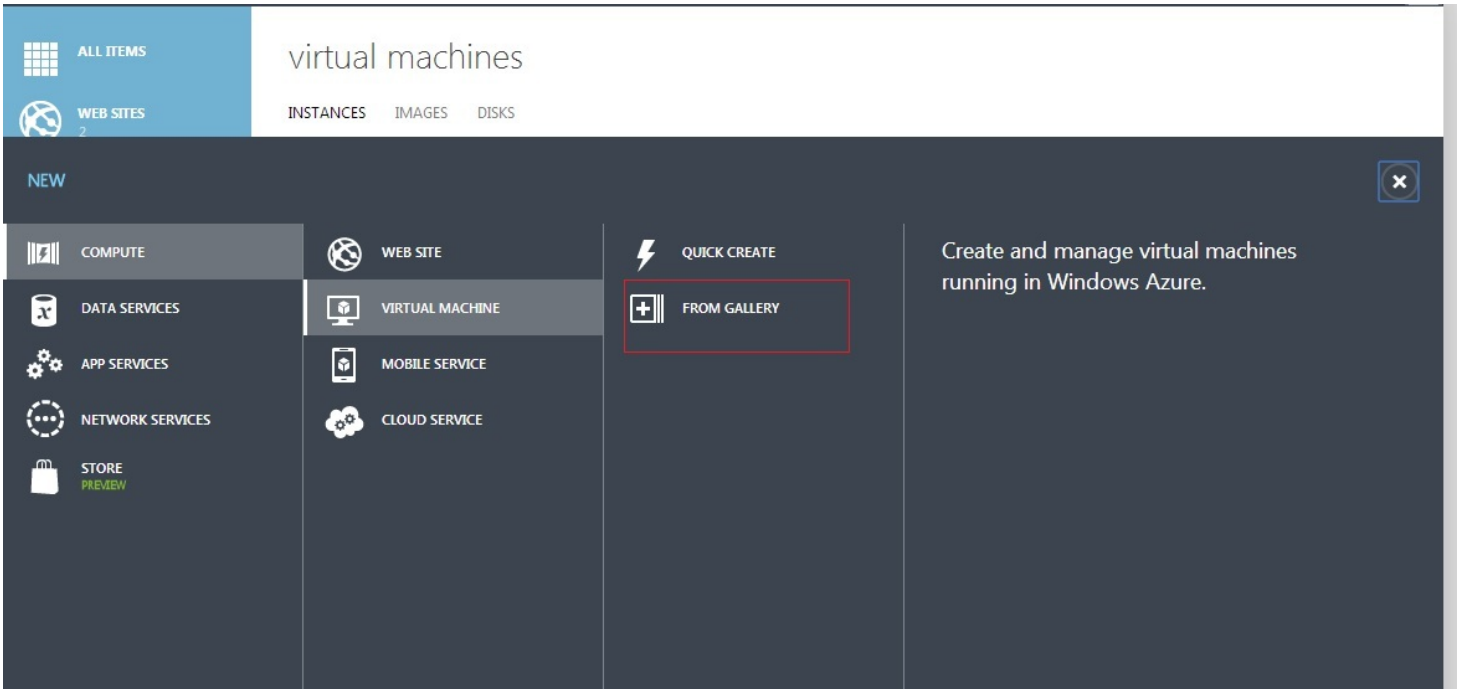


So, here as you can see in the left menu bar, I have already created one VM with Visual Studio 14. But, let me go ahead and show you the steps at least how to create the same. This is very simple, you

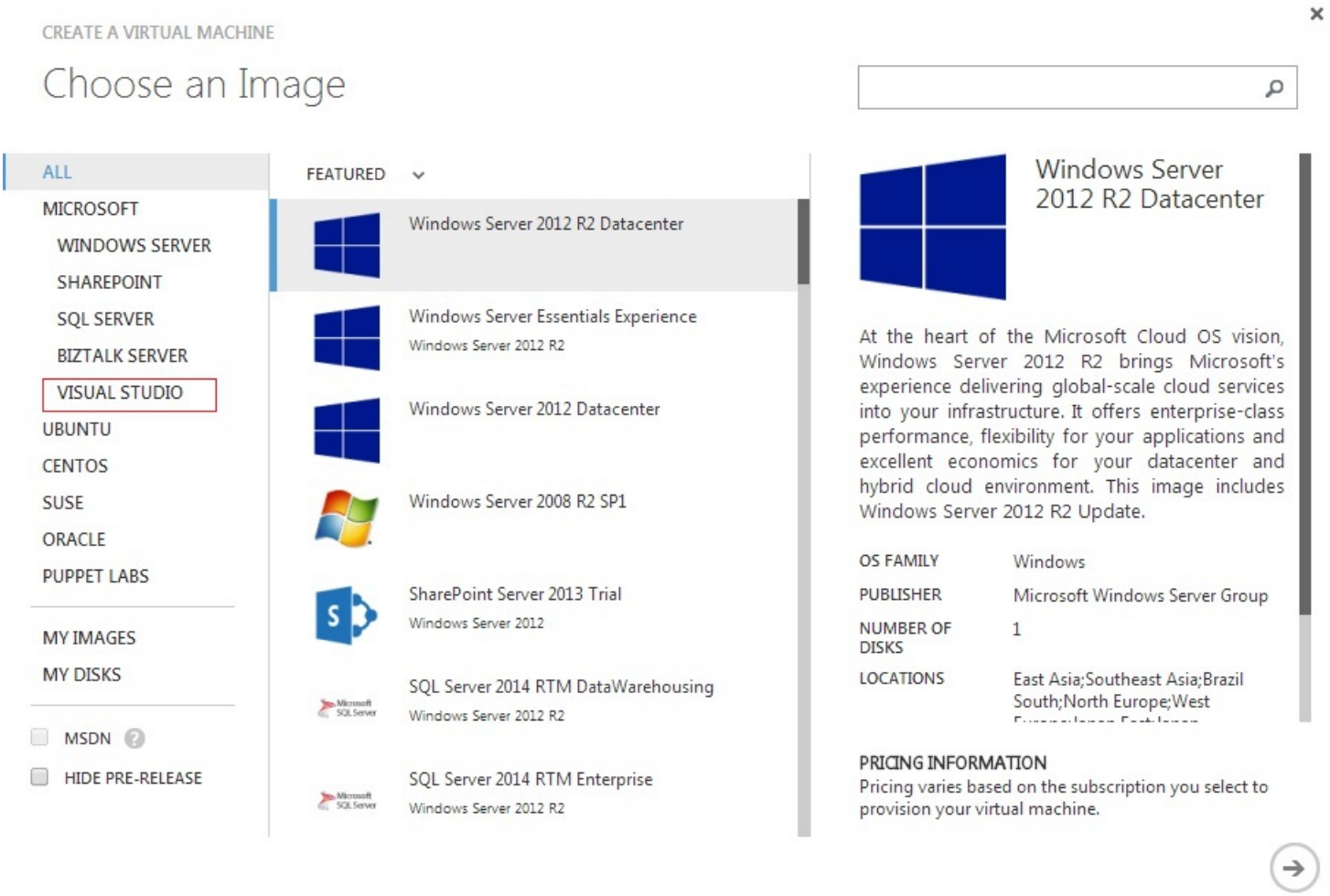
need to click on Virtual Machines link in the menu and then click on "***New***" button at the bottom of the page as shown below.



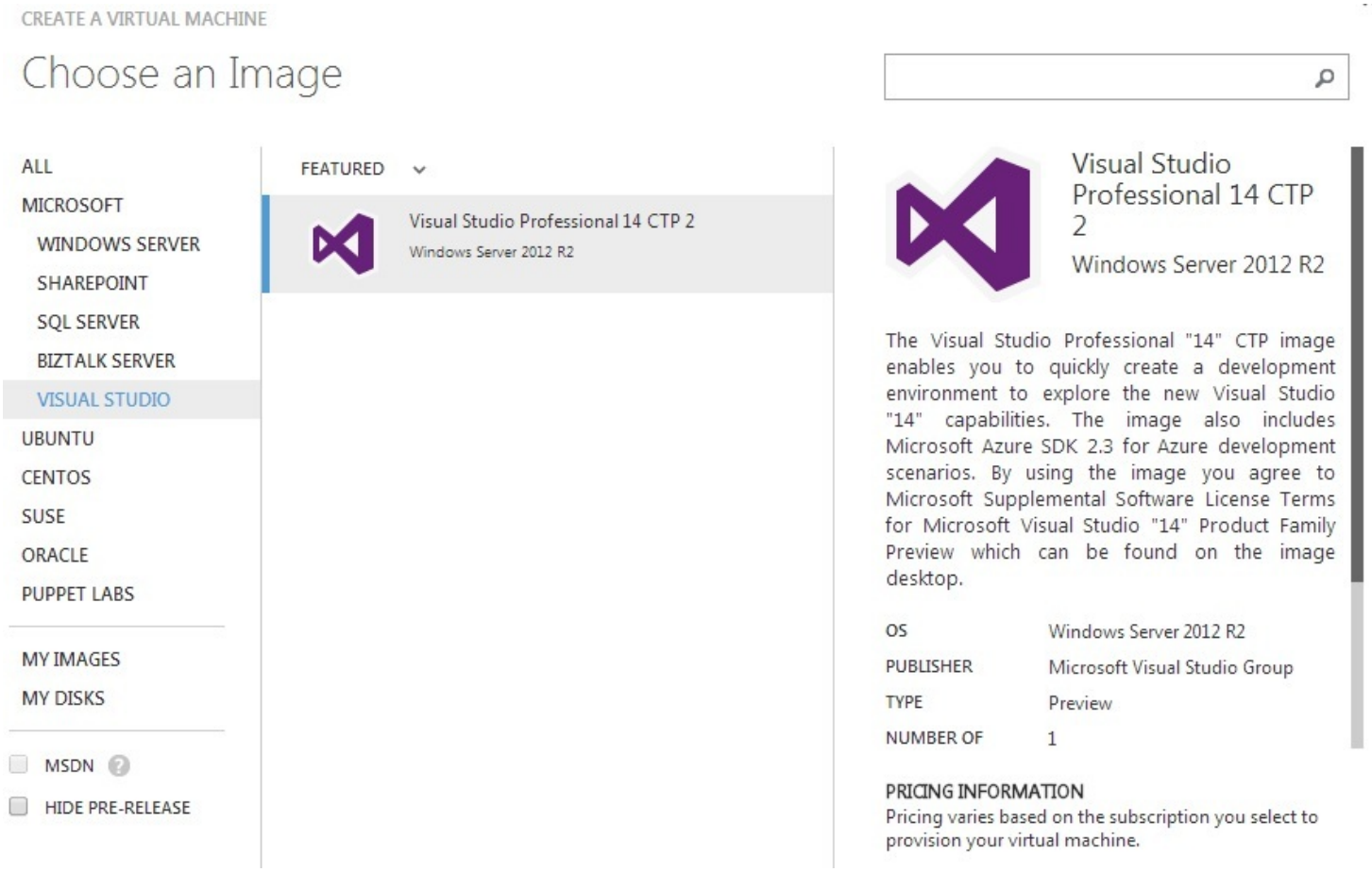
Now, as soon as you create on New button, it will open a new page with different options as shown below



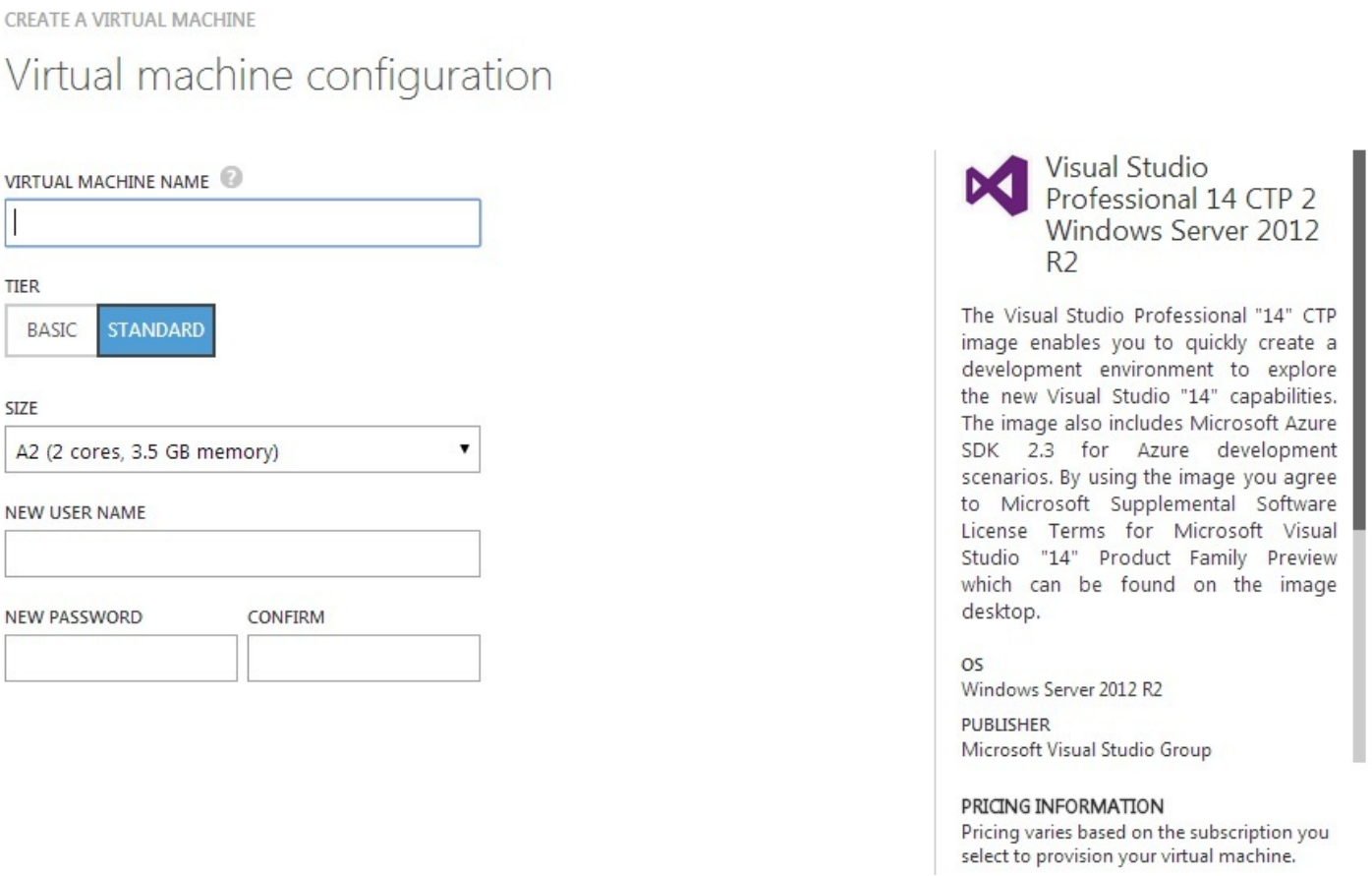
Then, I'll select create a VM "***From Gallery***". Then, this will again list me further options as shown below



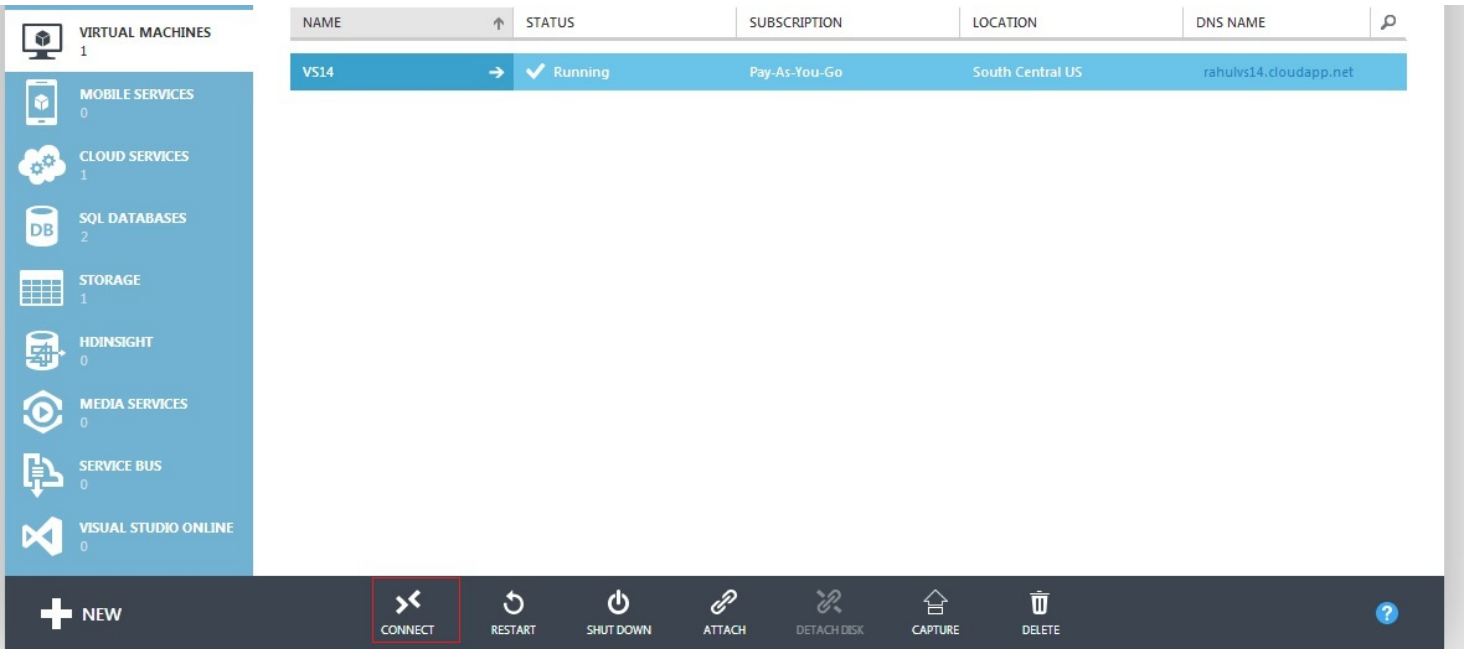
Now, it has listed me many options with which I can go ahead and create my Virtual Machine. However, in here I'll be selecting Visual Studio from the left pane as highlighted. Upon clicking on this, it will give me Visual Studio 14 option as shown ahead in the screen shot.



Now, when I click on next, then it will ask me usual details like username and password and some other details like VM configuration related stuffs as shown below in the screen shot.



Now, since I have already created my VM, so I'll close this here itself and login to my VM. In order to connect to the VM, you need to go to dashboard page, select the VM and then from the bottom of the page say connect as shown below in the screen shot.



Now, as soon as you click on this icon, it will download a utility to connect you to your remote machine. So, here this utility will ask for your username and password which you have provided by creating the VM. Upon Successful authentication, it will launch the Virtual Machine as shown below in the screen shot.

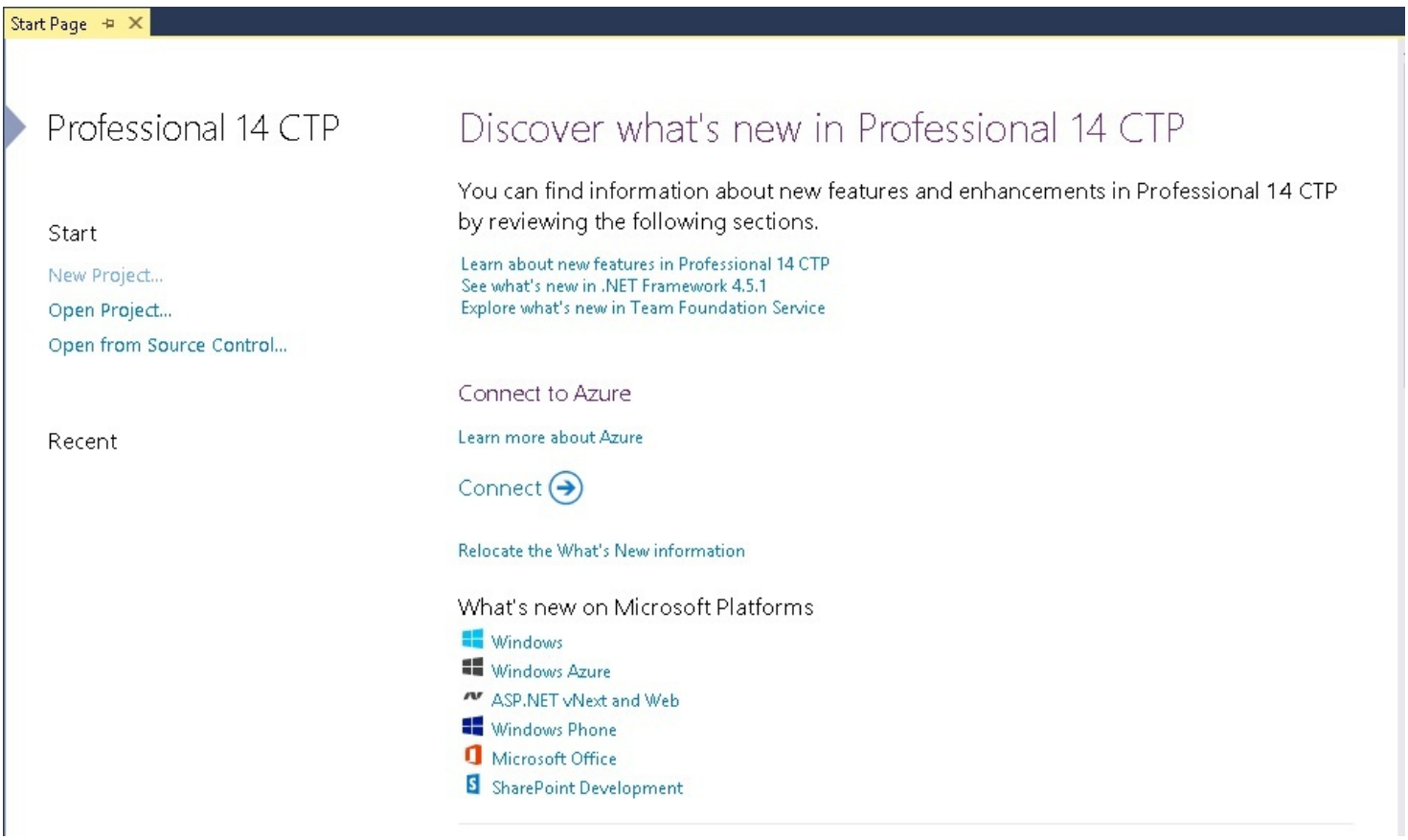


Now, here when you click on the desktop icon, it will show Visual Studio Icon on the desktop itself as shown below in the screen shot.

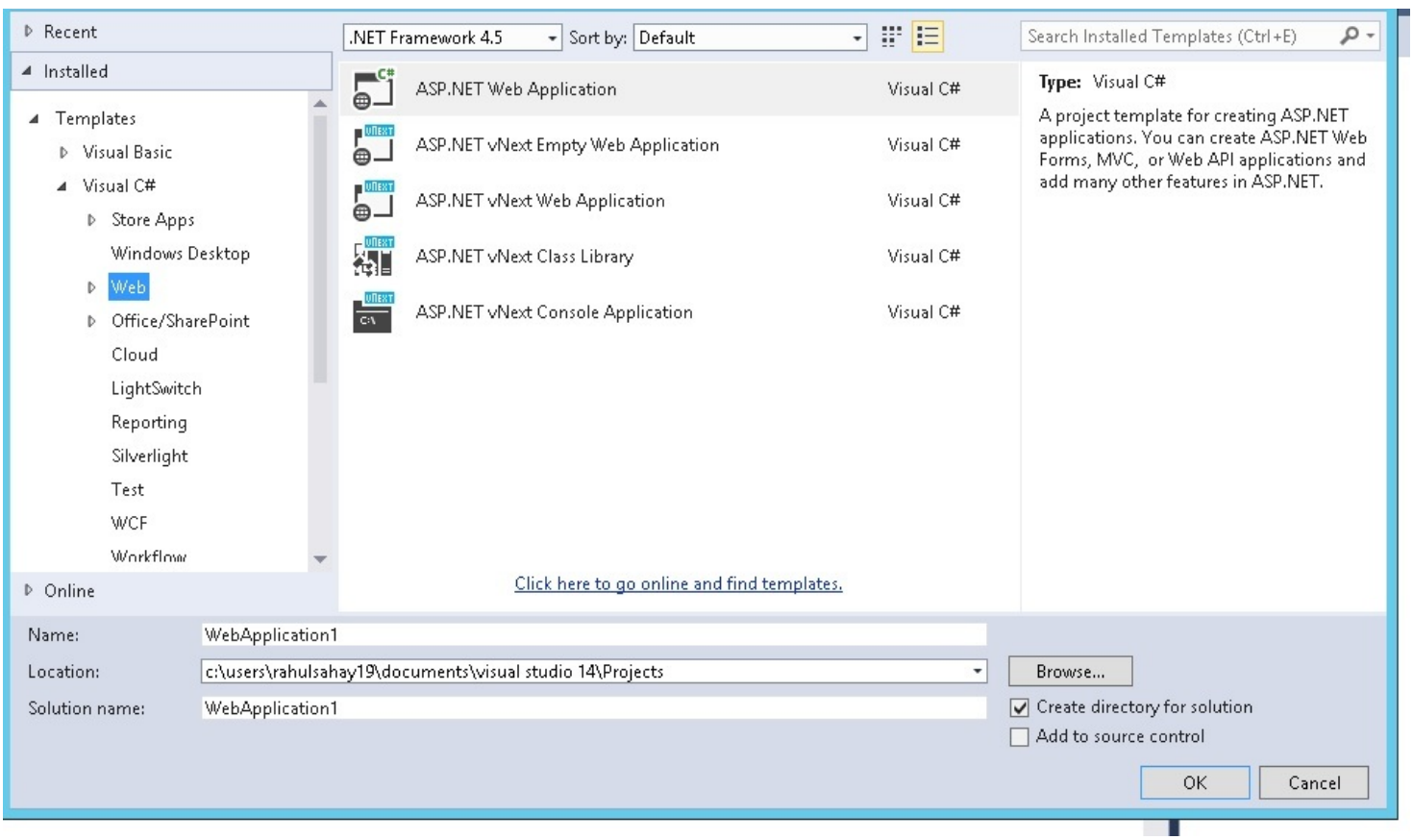


Getting Started with Visual Studio 14:-

So, now when you click on this icon it will launch the new version of the Visual Studio with ASP.Net vNext as shown below in the screen shot.



Now, when you click on the New Project, it will launch template selector screen and from that under that C# category, when you select web as shown in the below screen shot, it will list **vNext** templates as shown below.



So, here as you can see that, currently all different kinds of vNext templates are

listed under the same web stack. This organization of templates will change once full-fledged version of **ASP.NET vNext** is released. And also it is quite possible that name "***vNext***" will also get changed with some other more meaningful name. Now, let's discuss different templates listed here in brief.

- **ASP.Net Web Application** :- I don't think for this particular template I have to say anything as this is the same old template which we have been using earlier in VS 2013, where in it is popping an option selector window to select a MVC, Web Forms, Web API …… template.
- **ASP.Net vNext Empty Web**

Application: - Again this is very obvious thing. This is an empty app for professional who want to write each bit on their own right from the scratch.

- **ASP.Net vNext Web Application**: - Now, this is full blown app. Very good starting point for all developers. It has features like **Bootstrap**, **ASP.Net Identity** etc. So, basically this template will have all the basic stuffs needed to jump start with a new application.
- **ASP.Net vNext Class Library**: - Class library is again a very powerful and familiar concept wherein we are building

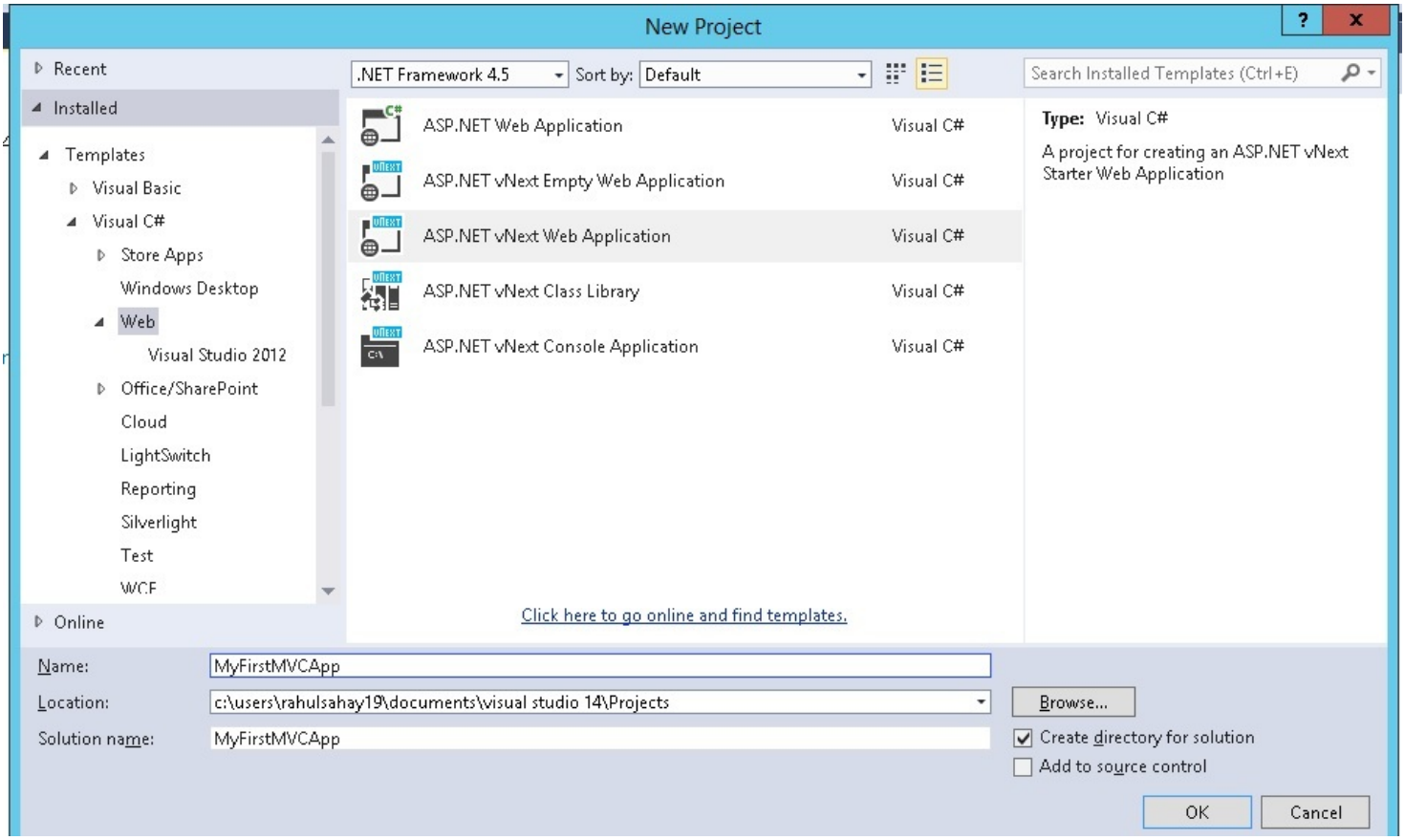
reusable components under one repository. Now, the special thing about this class library is; it will act as plugin. It is going to plugin in ASP.Net vNext Loader mechanism, which I'll show later in this chapter.

- **ASP.Net vNext Console Application**: - So, if you remember earlier as I said ASP.NET vNext can be self-hosted as well. So, you can think of this vNext Console App as self-hosted project template.

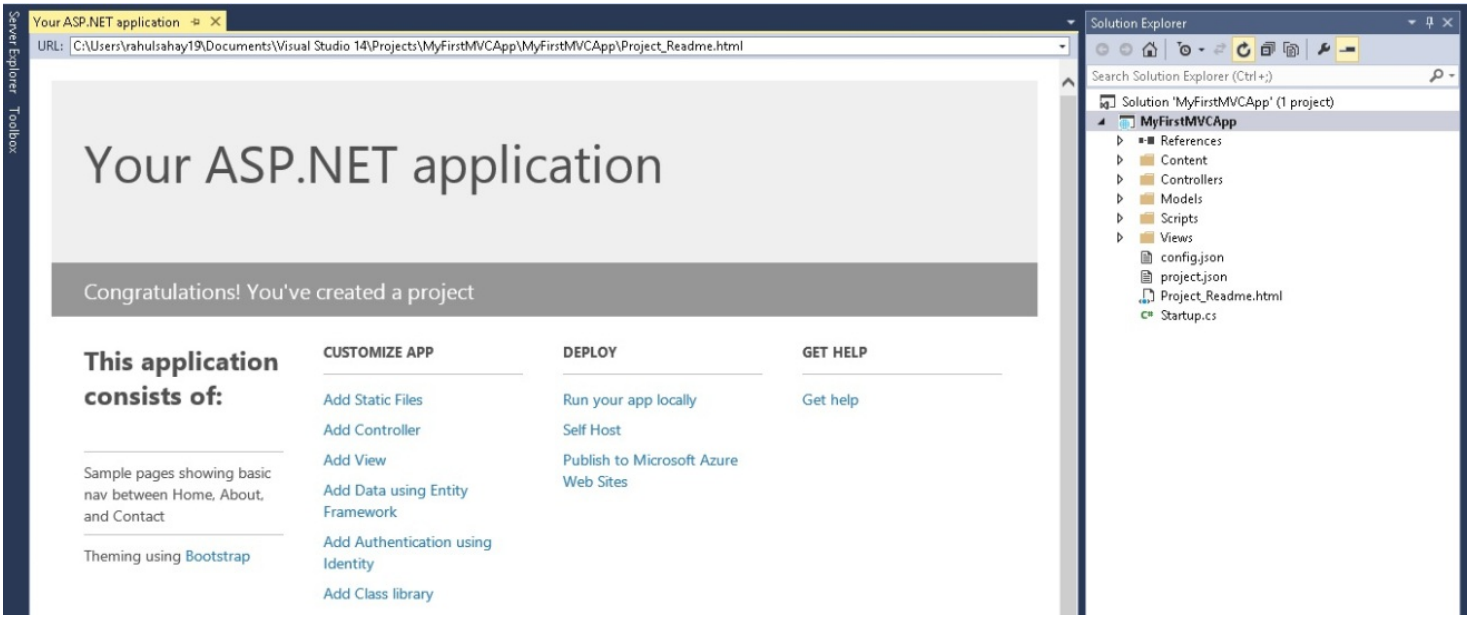
Creating your 1st Visual Studio 14 MVC

Project:-

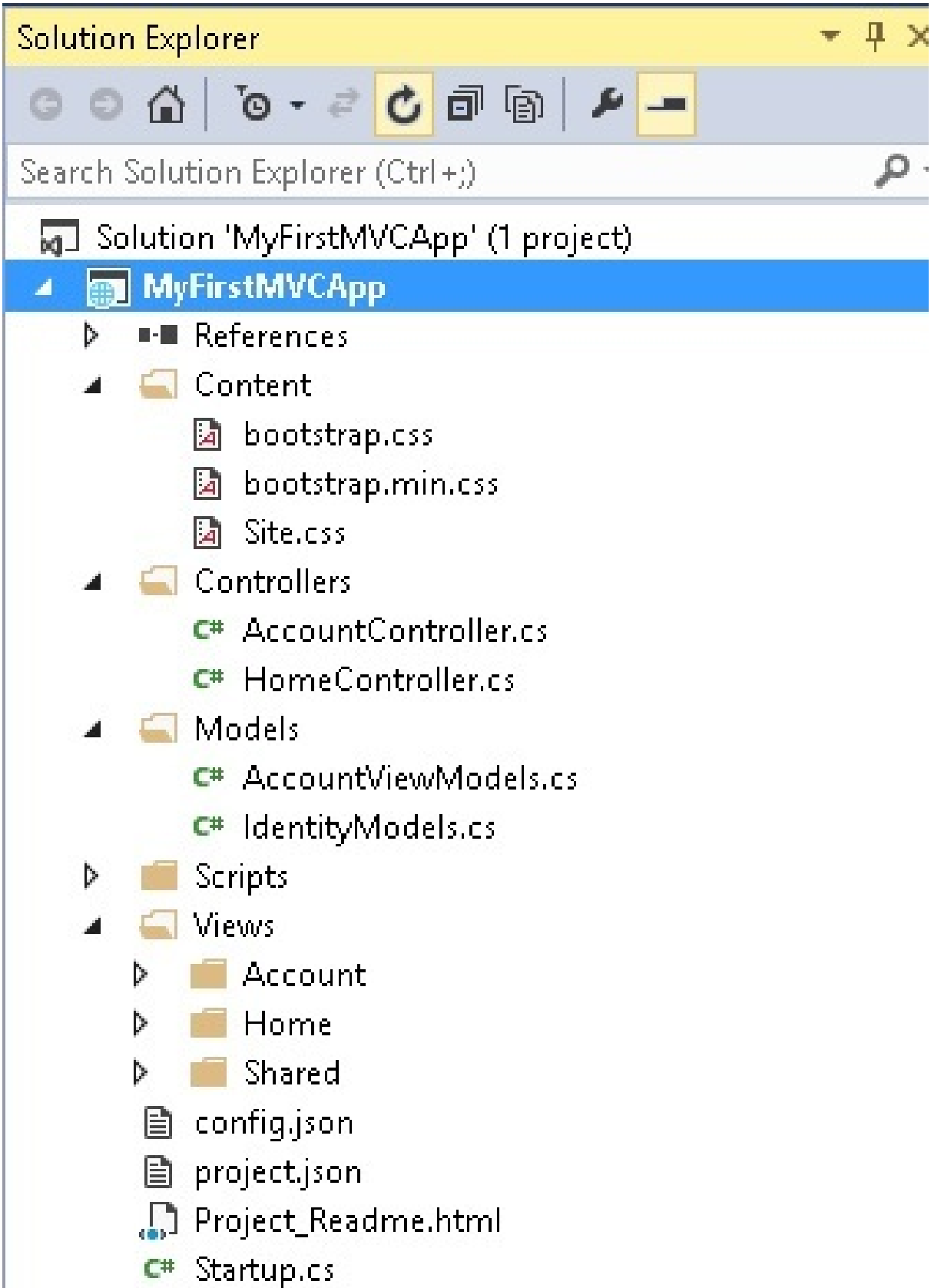
So, let's go ahead and create my 1st MVC project and let me give it name "MyFirstMVCApp".



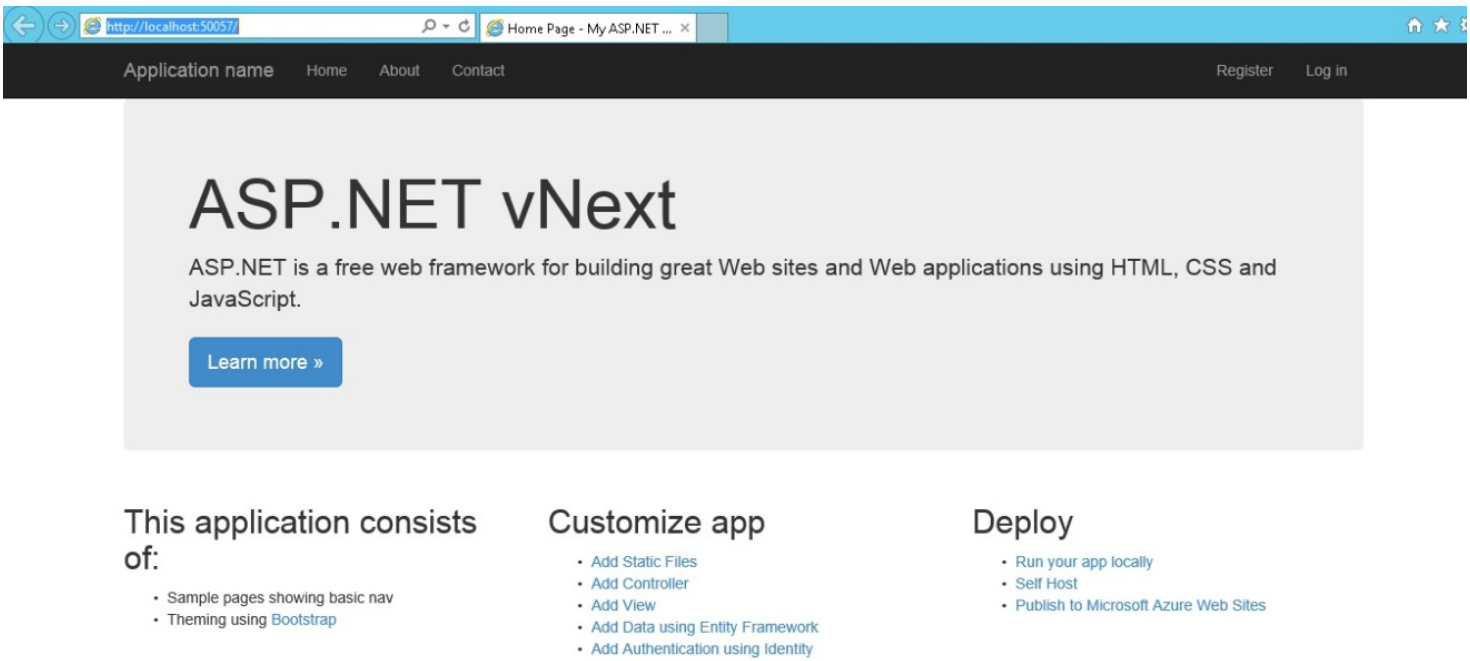
So, below is the screen shot of the default application.



So, now when you expand different folders in solution explorer, you will see very much MVC stuffs what we have done earlier.

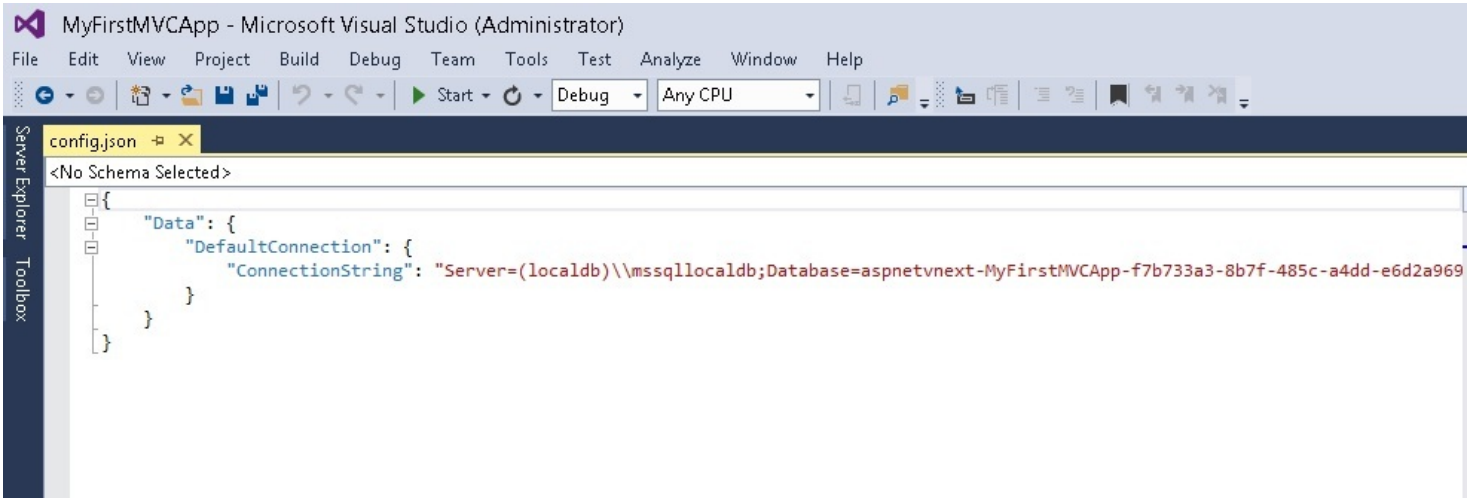

Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'MyFirstMVCApp' (1 project)
MyFirstMVCApp
References
Content
bootstrap.css
bootstrap.min.css
Site.css
Controllers
AccountController.cs
HomeController.cs
Models
AccountViewModels.cs
IdentityModels.cs
Scripts
Views
Account
Home
Shared
config.json
project.json
Project_Readme.html
Startup.cs

Now, before delving deep inside, let's try running this app at once 1st.



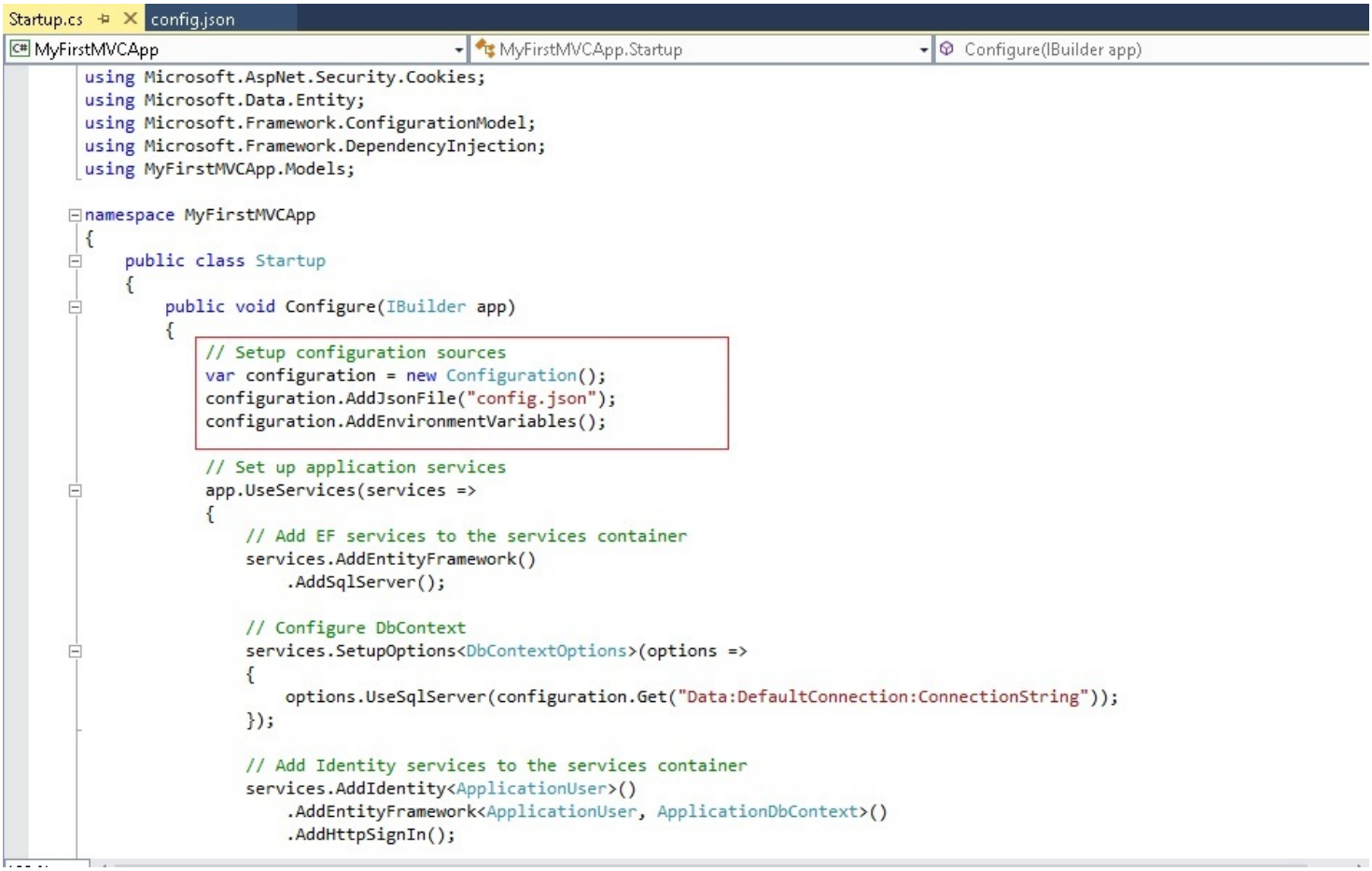
So, almost all the things are similar, what we usually have in ASP.NET MVC. But, some of the different things which you can easily pick up simply scanning the solution explorer like "***config.json***".

Configurations in MVC Project:-



So, with ASP.Net vNext Microsoft created new config system called as "***config.json***". So, basically here this template keeps track of application settings such as connection strings as shown above in the screen shot. In a vNext project we can use a "***Web.config***" file for application settings, but we have other options as

well. For example, we can plug in different NuGet configuration packages that enable you to use "***.ini***" files or "***.json***" files. The template uses "***Microsoft.Framework.Configuration***" and creates a "***config.json***". The Startup class includes this code that gets the settings from "***config.json***" and puts them in environment variables:



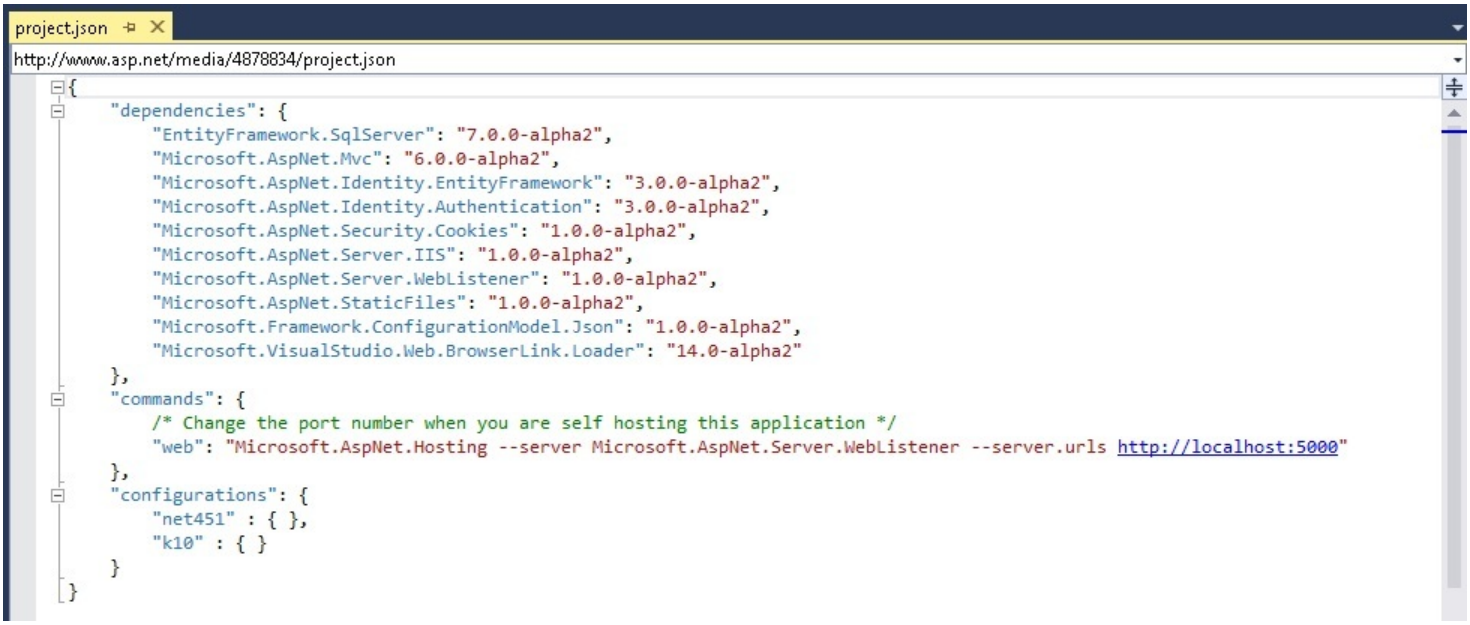
Now, the "***Configuration***" class also provides other methods for working with configuration data, such as a "***Get***" method that enables you to retrieve a specified setting as shown below

```
// Setup configuration sources
var configuration = new Configuration();
configuration.AddJsonFile("config.json");
configuration.AddEnvironmentVariables();

string connectionString = configuration.Get("Data:DefaultConnection:ConnectionString");
```

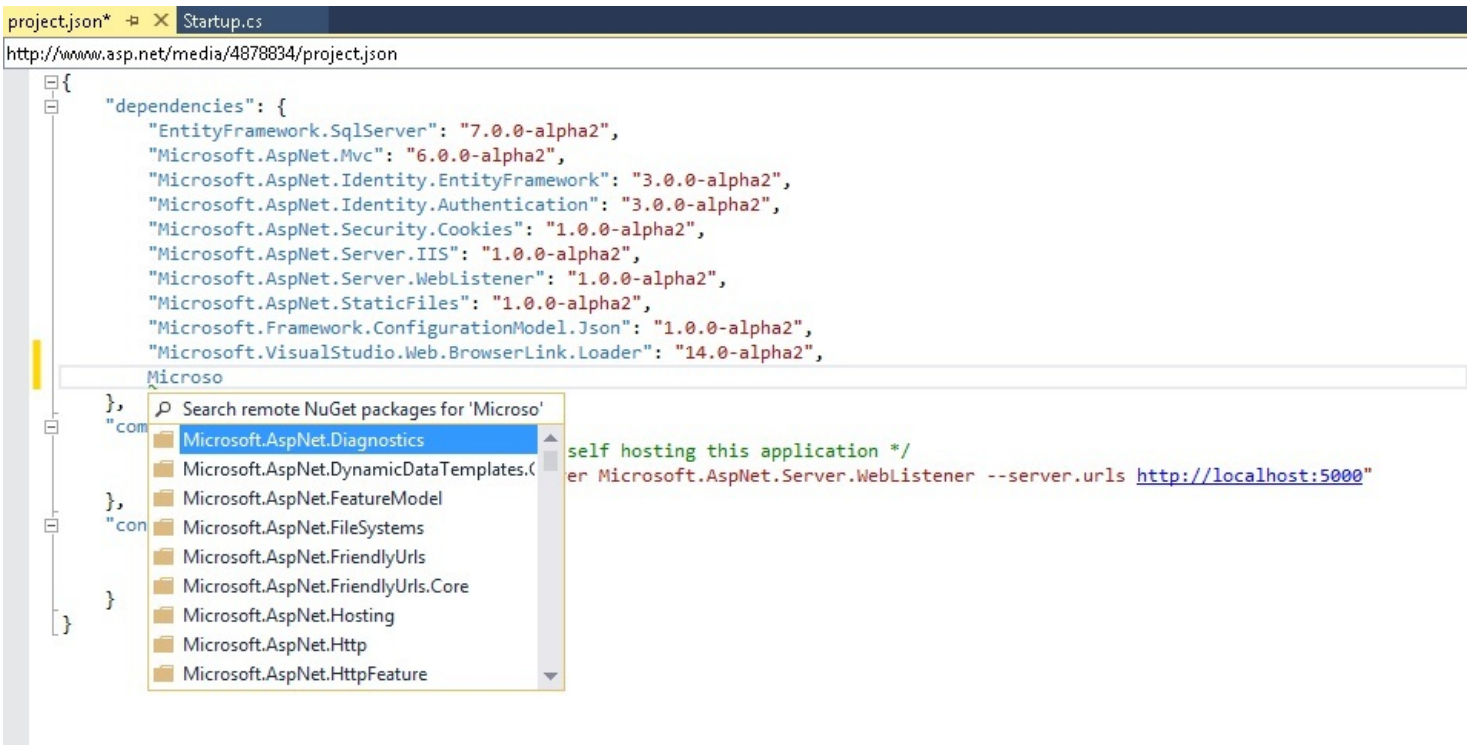
Project.json:-

"***Project.json***" is the file which maintains a list of dependencies for the project and also list of build out configurations. It also includes list of commands. As shown below in the screen shot, first part is list of dependencies on which project is built.



Here, in this project.json you can also

get intellisense. Like suppose you want to add any new dependency. You can directly add from here as well with fully qualified version no as shown below.



So, as you can see in the above screen shot, it will search in nuget package manager will try to get the exact package which you are trying to resolve. Then, it has a list of commands which is going to

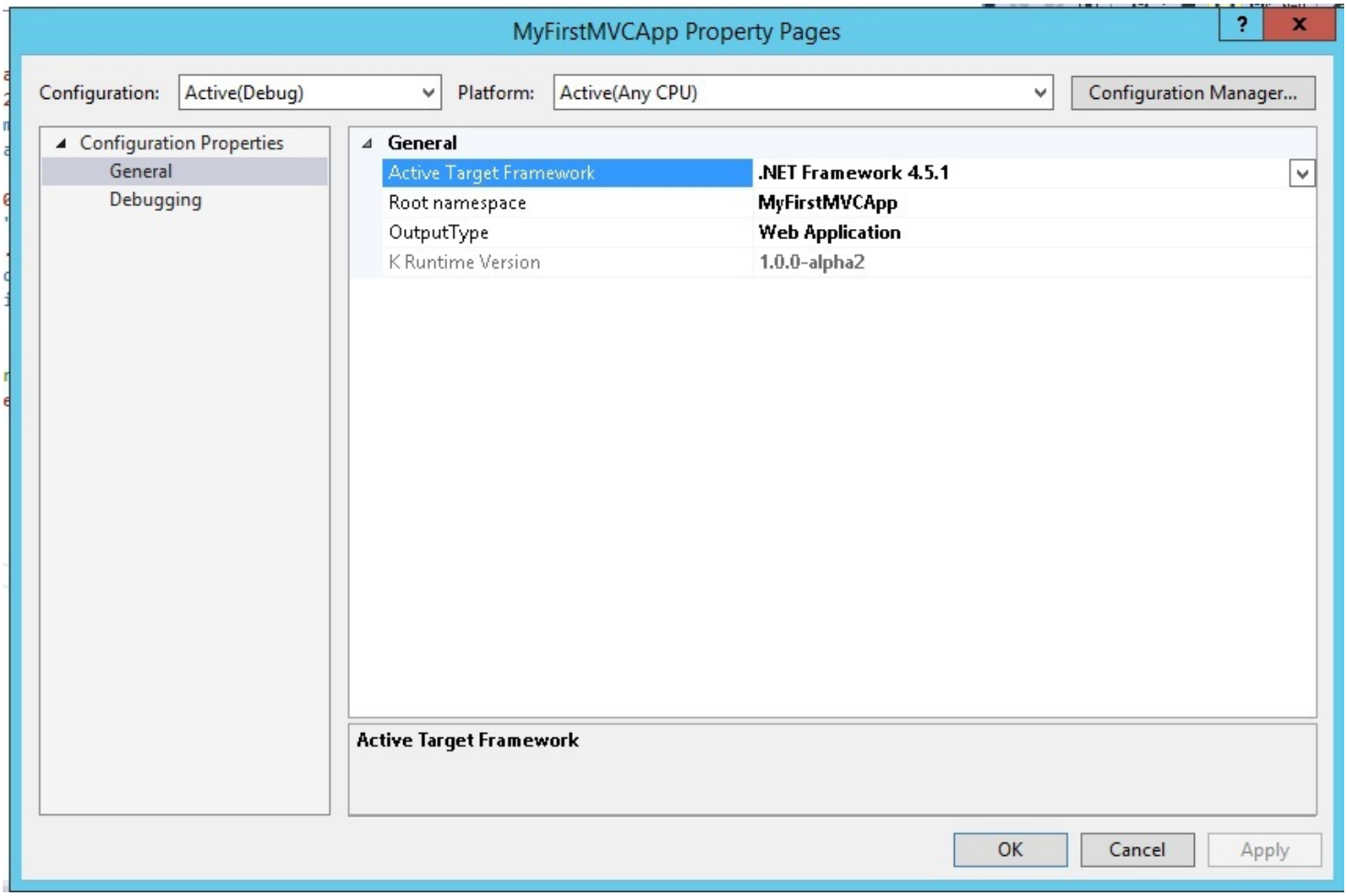
be used when you are going to host the application on its own basically self-hosting.

```
"commands": {
    /* Change the port number when you are self hosting this application */
    "web": "Microsoft.AspNet.Hosting --server Microsoft.AspNet.Server.WebListener --server.urls http://localhost:5000"
},
```

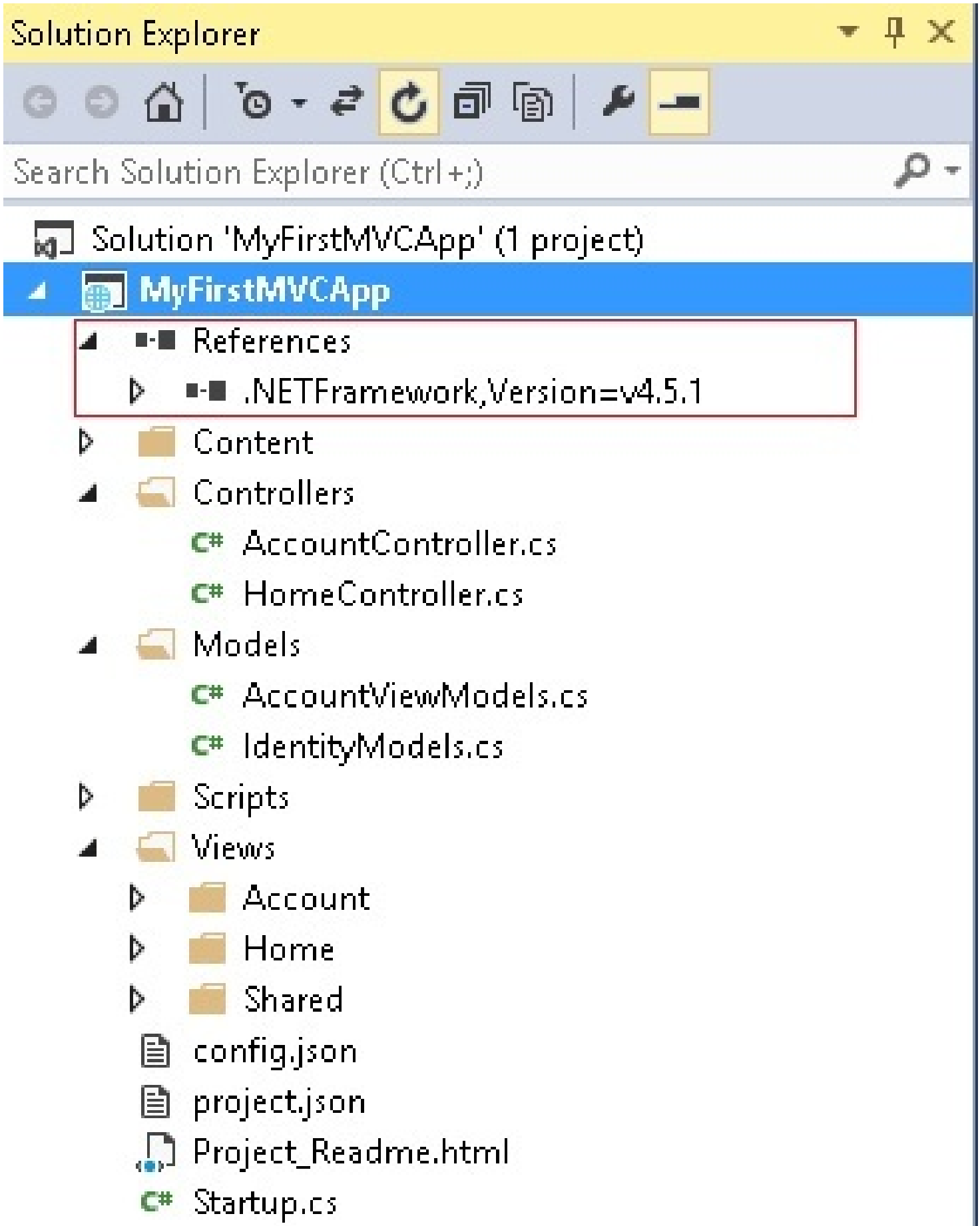
Then, in project.json file we also have build output configurations options as shown below.

```
"configurations": {
    "net451" : { },
    "k10" : { }
}
```

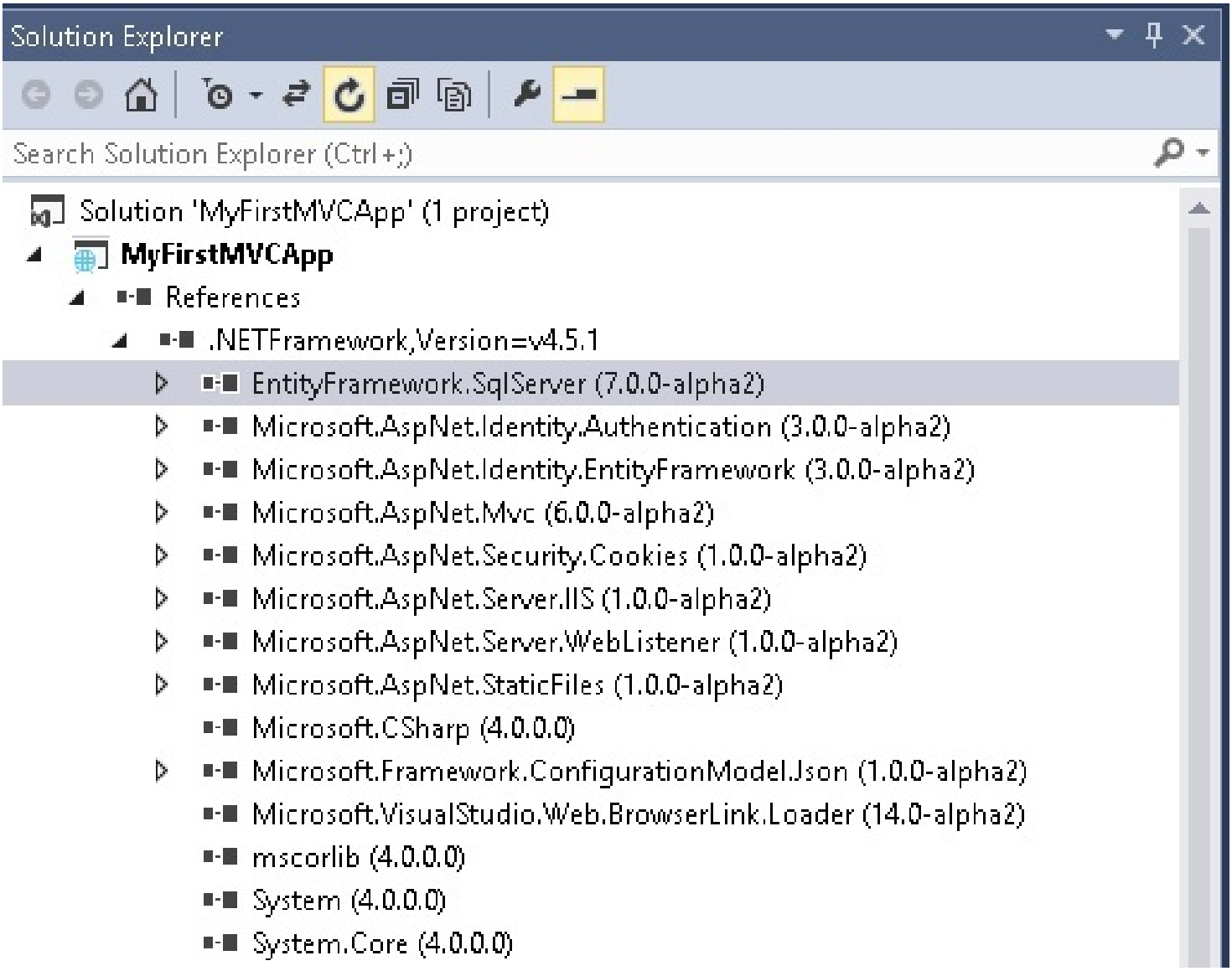
Now, these settings correspond to "***Active Target Framework***" options in the property pages as shown below.



Now, here "***Active Framework***" is set to "***4.5.1***". Now, if you see references for this project, it will be different from earlier versions of .NET. Now, it is more tidy neat and clean

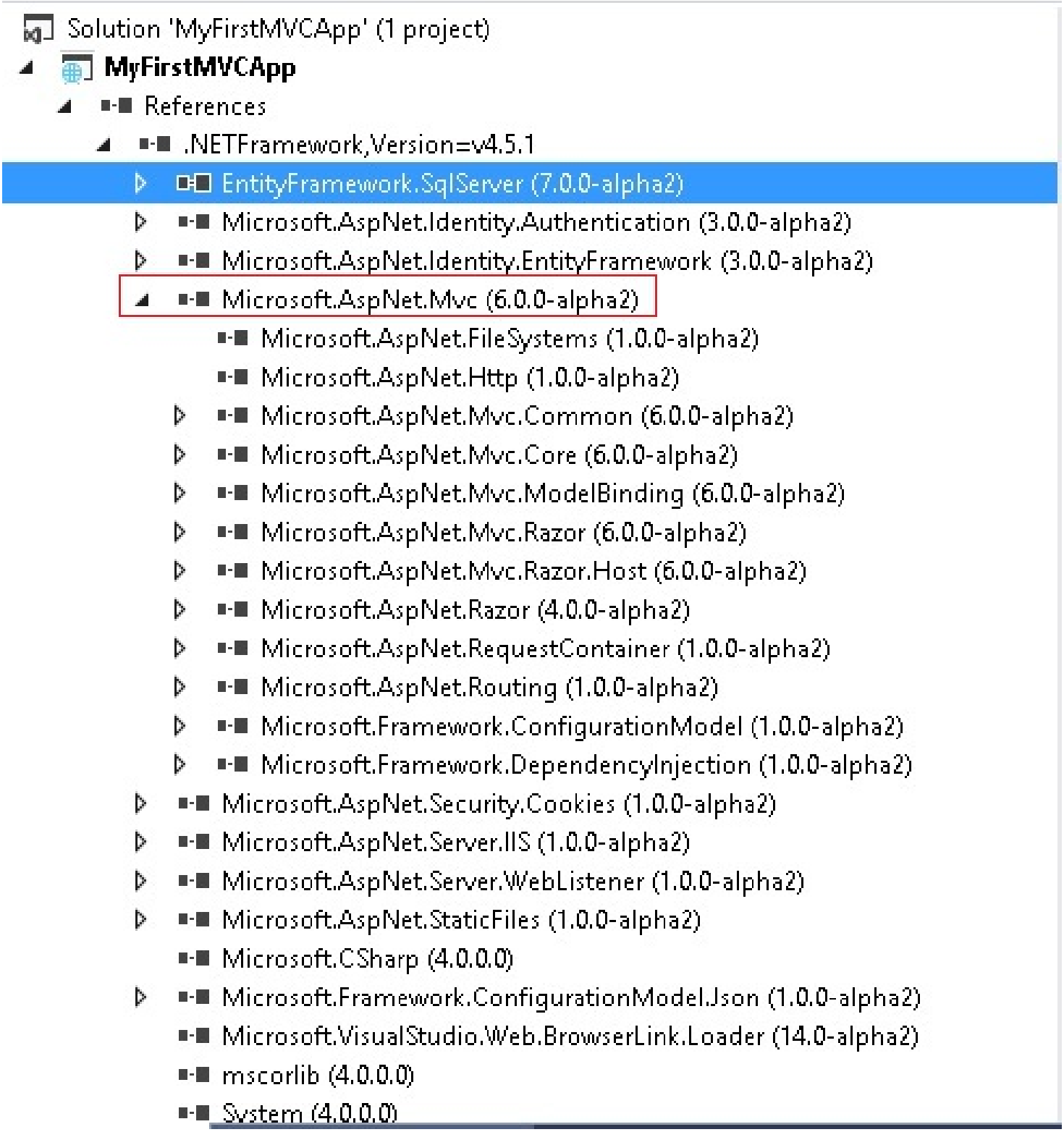

Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'MyFirstMVCApp' (1 project)
MyFirstMVCApp
References
.NETFramework,Version=v4.5.1
Content
Controllers
AccountController.cs
HomeController.cs
Models
AccountViewModels.cs
IdentityModels.cs
Scripts
Views
Account
Home
Shared
config.json
project.json
Project_Readme.html
Startup.cs

But, when you collapse this one, then it will list all the project related dependencies



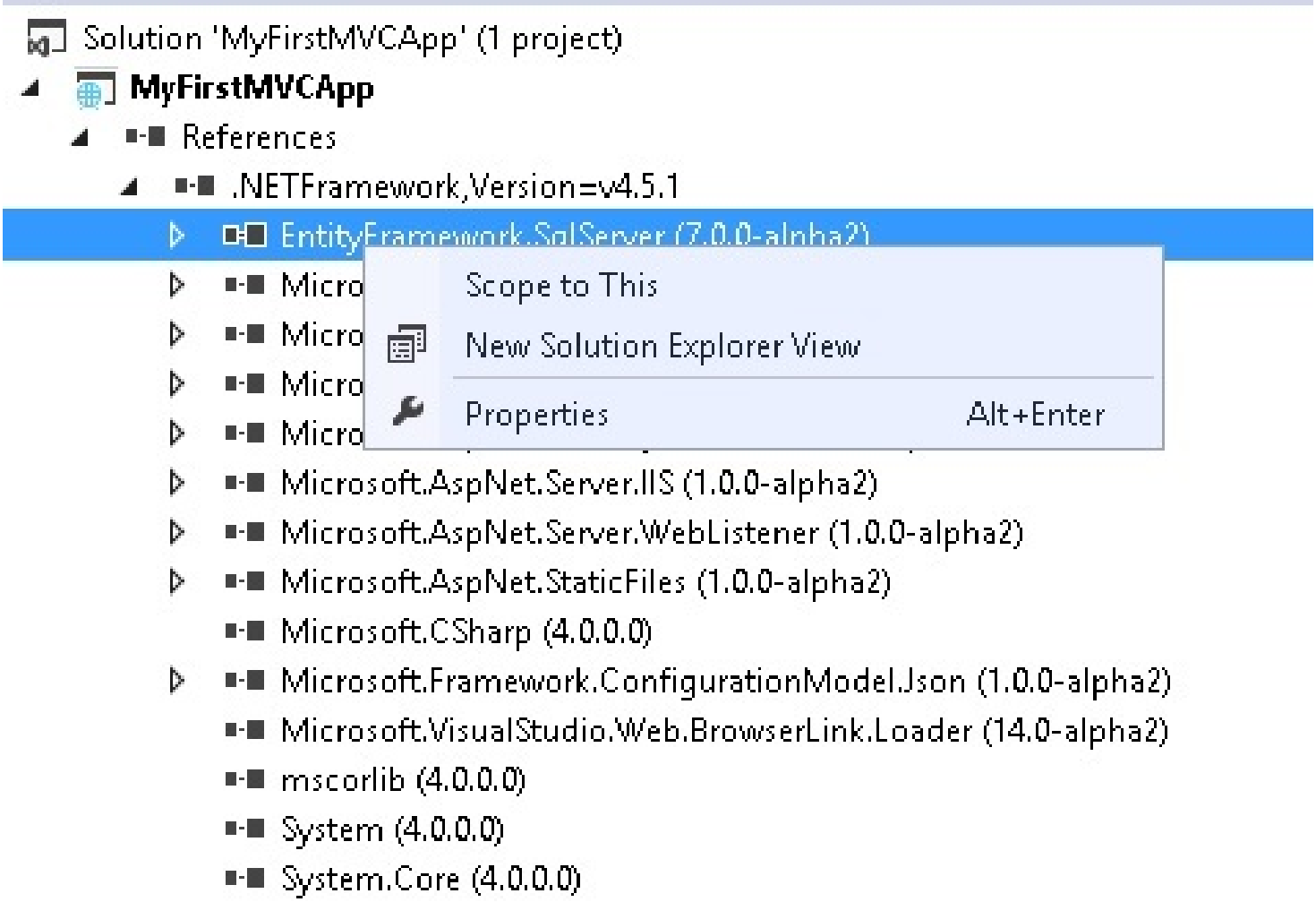
So, here the idea is list out dependency tree. So, all meaningful and

understandable part is listed above and when you expand these nodes, you will get more children for this parent node as shown below in the screen shot.

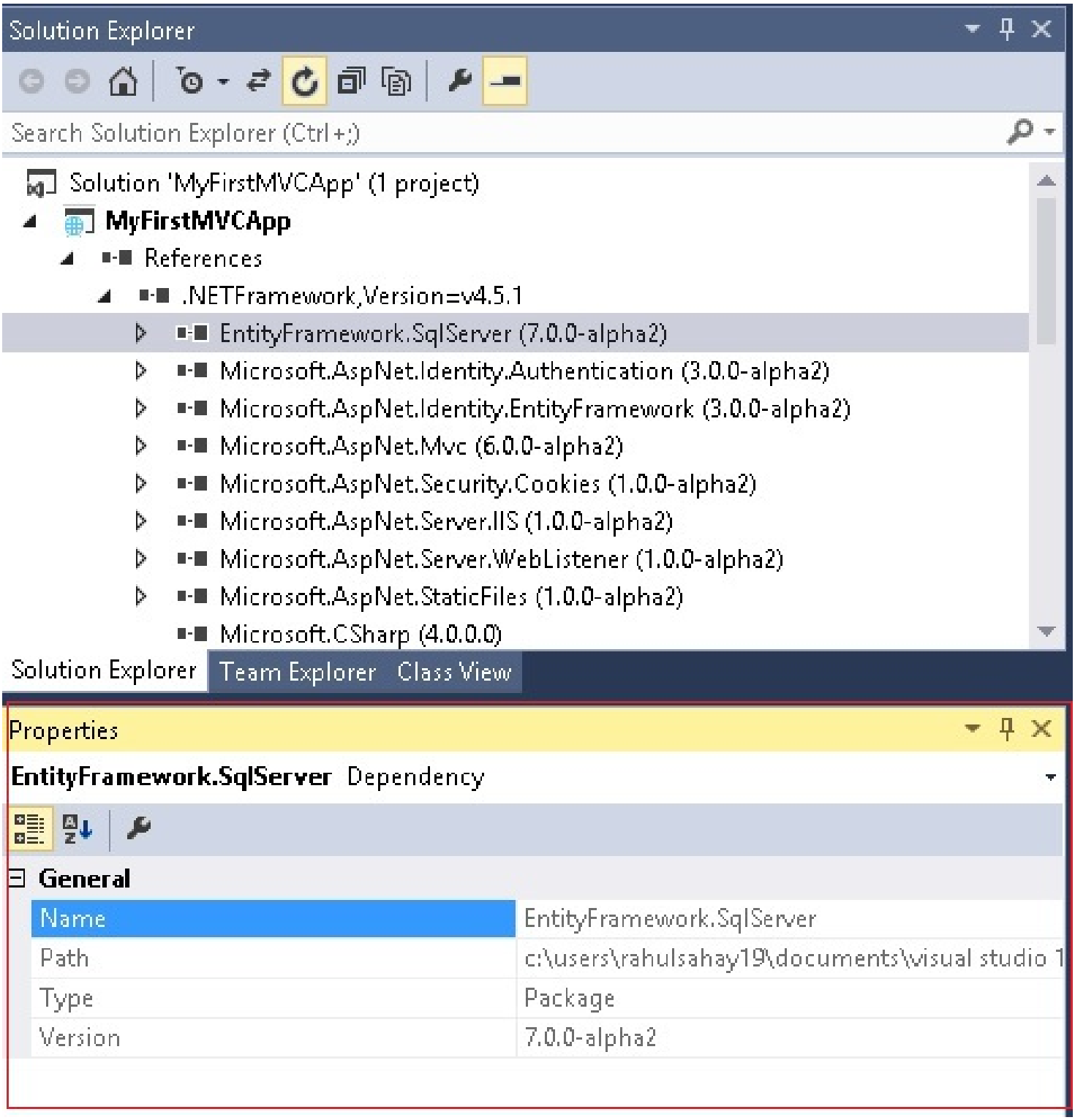


So, when you have to see property for any of these references, you can just

right click that check it's properties like explained below.



When you click on "***Properties***" for that, it will tell exactly where the "***dll***" is located and its version



Startup class:-

So, this is the place where everything is configured. If it is not here, it means it's not there in the application. And one thing which you might have noticed in the startup file that there is no "***System Web***". So, if you need some kind of middleware kind of stuffs like **Static files**, **Cookie based authentication**, well you have to actually add in your application. By default, the vNext hosting environment expects to find a startup class named "***Startup***". This class contains a "***Configure***" method that takes an "***IBuilder***" parameter. This configure method is used to configure HTTP pipelines. Now, let's focus what this template gives you by default

```
namespace MyFirstMVCApp
{
    public class Startup
    {
        public void Configure(IBuilder app)
        {
            // Setup configuration sources
            var configuration = new Configuration();
            configuration.AddJsonFile("config.json");
            configuration.AddEnvironmentVariables();

            string connectionString = configuration.Get("Data:DefaultConnection:ConnectionString");

            // Set up application services
            app.UseServices(services =>...);

            // Enable Browser Link support
            app.UseBrowserLink();

            // Add static files to the request pipeline
            app.UseStaticFiles();

            // Add cookie-based authentication to the request pipeline
            app.UseCookieAuthentication(new CookieAuthenticationOptions
            {
                AuthenticationType = DefaultAuthenticationTypes.ApplicationCookie,
                LoginPath = new PathString("/Account/Login"),
            });

            // Add MVC to the request pipeline
            app.UseMvc(routes =>...);
        }
    }
}
```

We have environment based config system which can pull name value pairs from variety of sources. So, it can pull values from "**json**" files, it can also pull environment variables and many other things.

```
// Set up application services
app.UseServices(services =>
{
    // Add EF services to the services container
    services.AddEntityFramework()
        .AddSqlServer();

    // Configure DbContext
    services.SetupOptions<DbContextOptions>(options =>
    {
        options.UseSqlServer(configuration.Get("Data:DefaultConnection:ConnectionString"));
    });

    // Add Identity services to the services container
    services.AddIdentity<ApplicationUser>()
        .AddEntityFramework<ApplicationUser, ApplicationDbContext>()
        .AddHttpSignIn();

    // Add MVC services to the services container
    services.AddMvc();
});
```

Now, as we discussed earlier in this chapter, that everything is going to be "***DI (Dependency Injection) first***" in MVC 6. So, "***UseServices***" is the implementation for that only. Now, here is the place where in you want to setup the services which you want to use across the application. Like here we have default setup for "***Entity Framework***", it's also configured

"***dbContext***" which will be injected into the controllers. Then, it's setting up "***ASP.Net Identity***" and last but not the least setting up the MVC Services.

Then, it gives you "***UseBrowserLink***", which enables you to edit your code and just save it and then upon refreshing in the browser it will take effect without building.

```
// Enable Browser Link support
app.UseBrowserLink();
```

Then, we have section for setting up static files in the middleware like "***Javascript files***".

```
// Add static files to the request pipeline
app.UseStaticFiles();
```

Then, we have cookie authentication

```
// Add cookie-based authentication to the request pipeline
app.UseCookieAuthentication(new CookieAuthenticationOptions
{
    AuthenticationType = DefaultAuthenticationTypes.ApplicationCookie,
    LoginPath = new PathString("/Account/Login"),
});

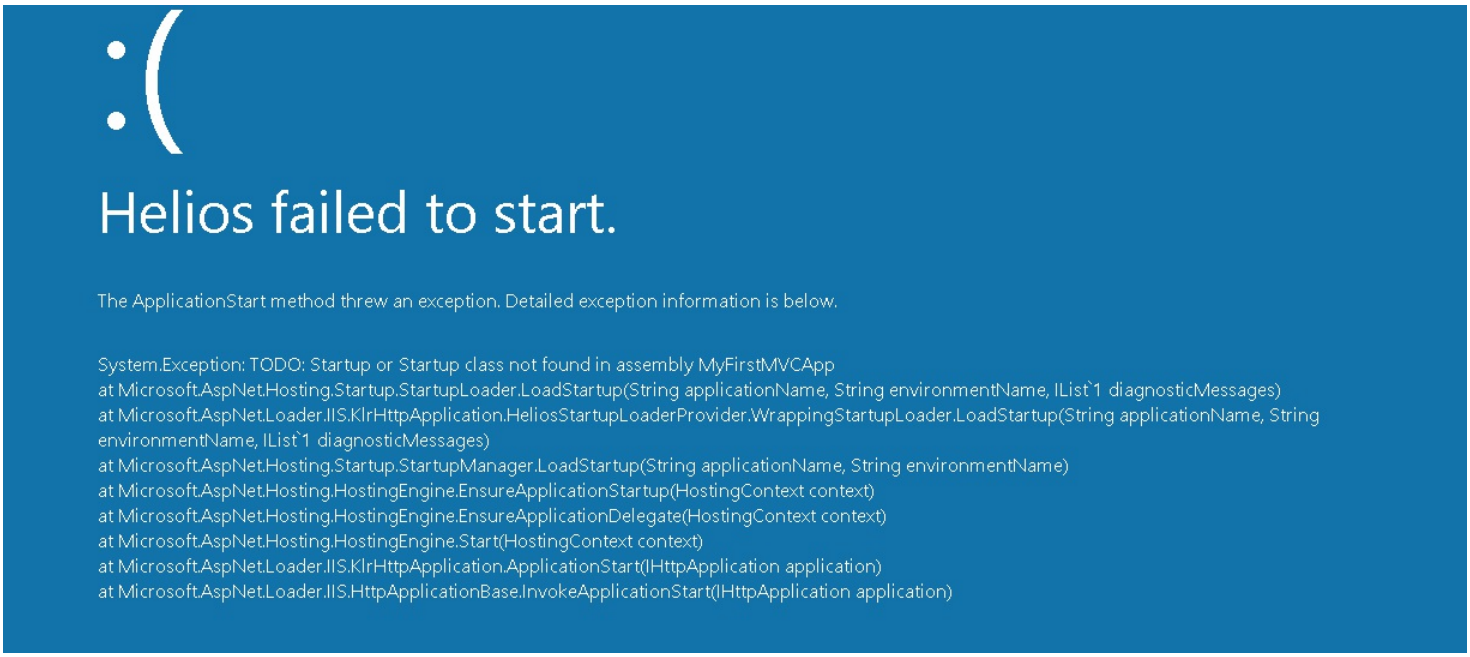
// Add MVC to the request pipeline
app.UseMvc(routes =>...);
```

And in the end we have routing declaration which is for both MVC and Web API.

```
        // Add MVC to the request pipeline
        app.UseMvc(routes =>
        {
            routes.MapRoute(
                name: "default",
                template: "{controller}/{action}/{id?}",
                defaults: new { controller = "Home", action = "Index" });

            routes.MapRoute(
                name: "api",
                template: "{controller}/{id?}");
        });
    }
```

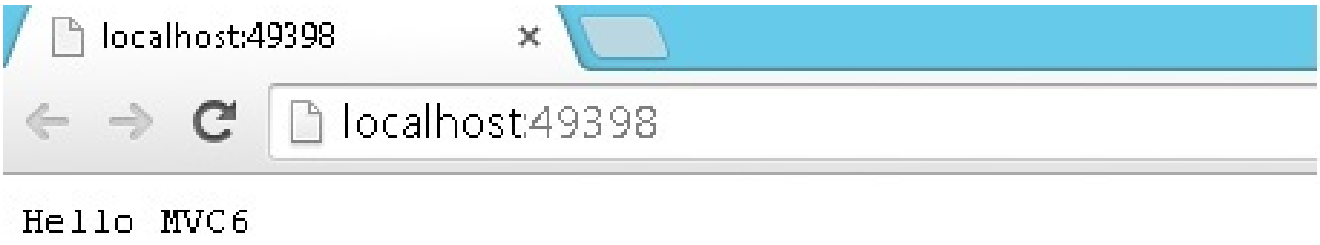
Now, let's look Start Up class in detail. For an instant, let me go ahead and comment out the complete "***Startup.cs***" class and then refresh the page. So, now going ahead you don't need to compile the solution every time for the change. So, in MVC 6 you just save the changes and F5 in browser, then it will take the effect in the browser. Thanks to new "***Roslyn compiler***". So, now when I refresh the page, it will produce me the below output.



So, now the another fancy thing is that "***Yellow Screen of Death***" is replaced with new fancy screen as shown above with complete stack trace. So, now let me go ahead and try something different, let's try build an **Async** request. Now, one more important point to note here in MVC 6, everything is going to be Async. So, what this will do, it will take the Async request and write to the response.

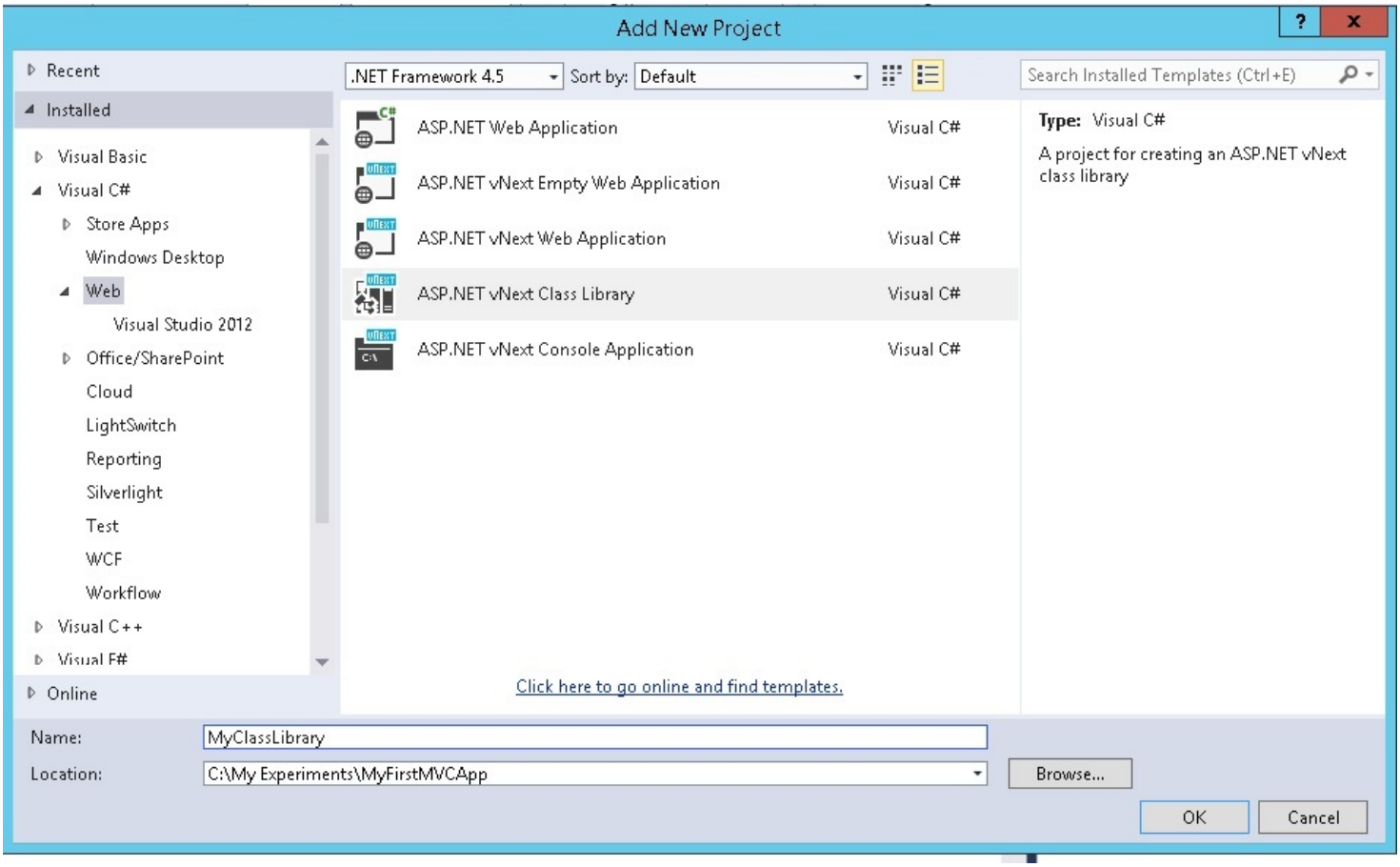
```
app.Run(async req => await req.Response.WriteAsync("Hello MVC6 "));
```

And, this will produce me the below output

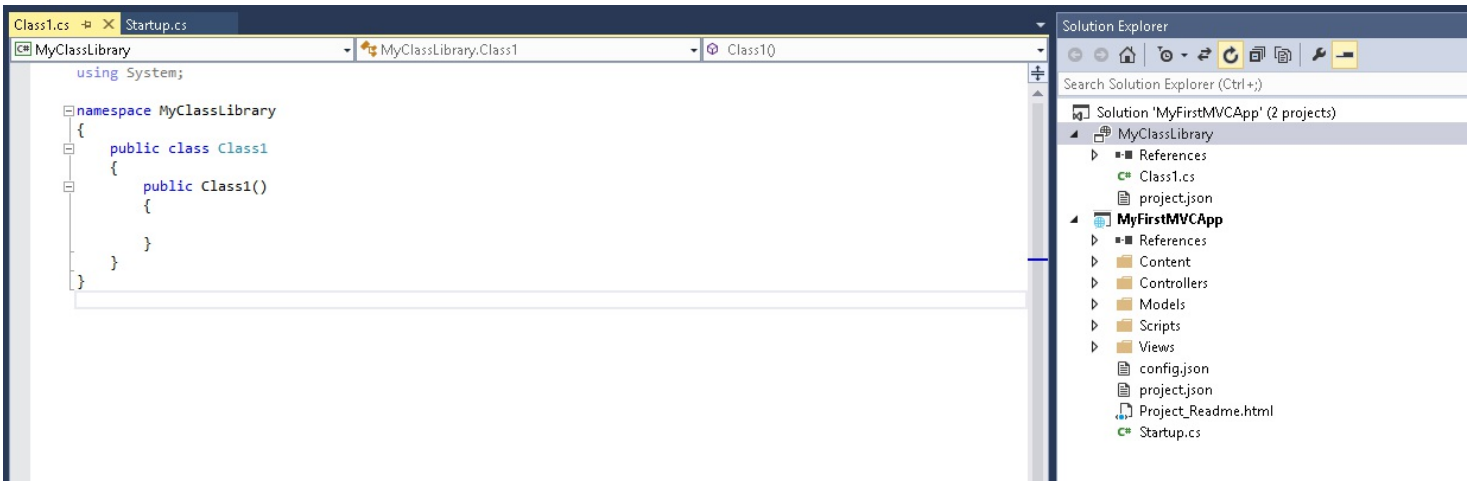


Adding a Class Library Project:-

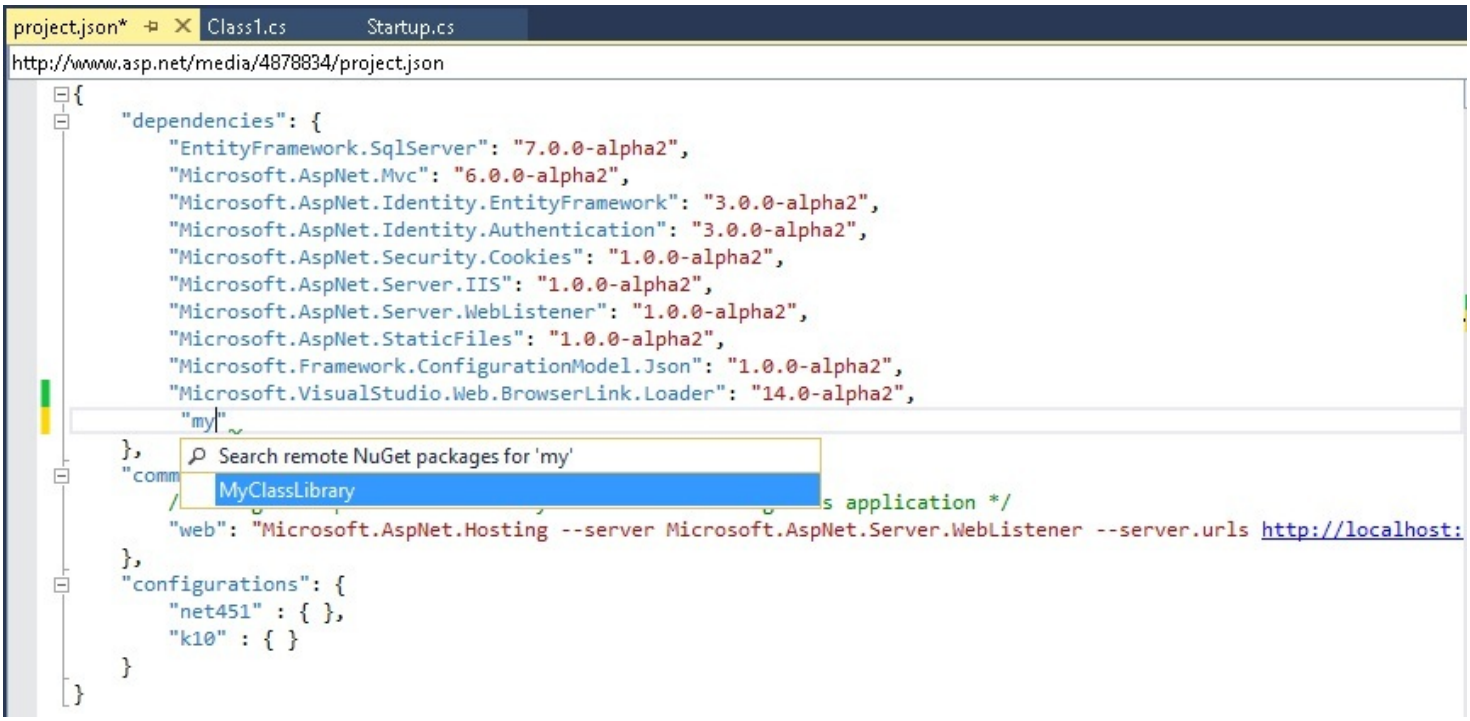
Now, let's go ahead and try adding another type of project in our solution. I gave it name "***MyClassLibrary***".



Now, once it gets added successfully. It will be presented in the default behavior, so no change here.

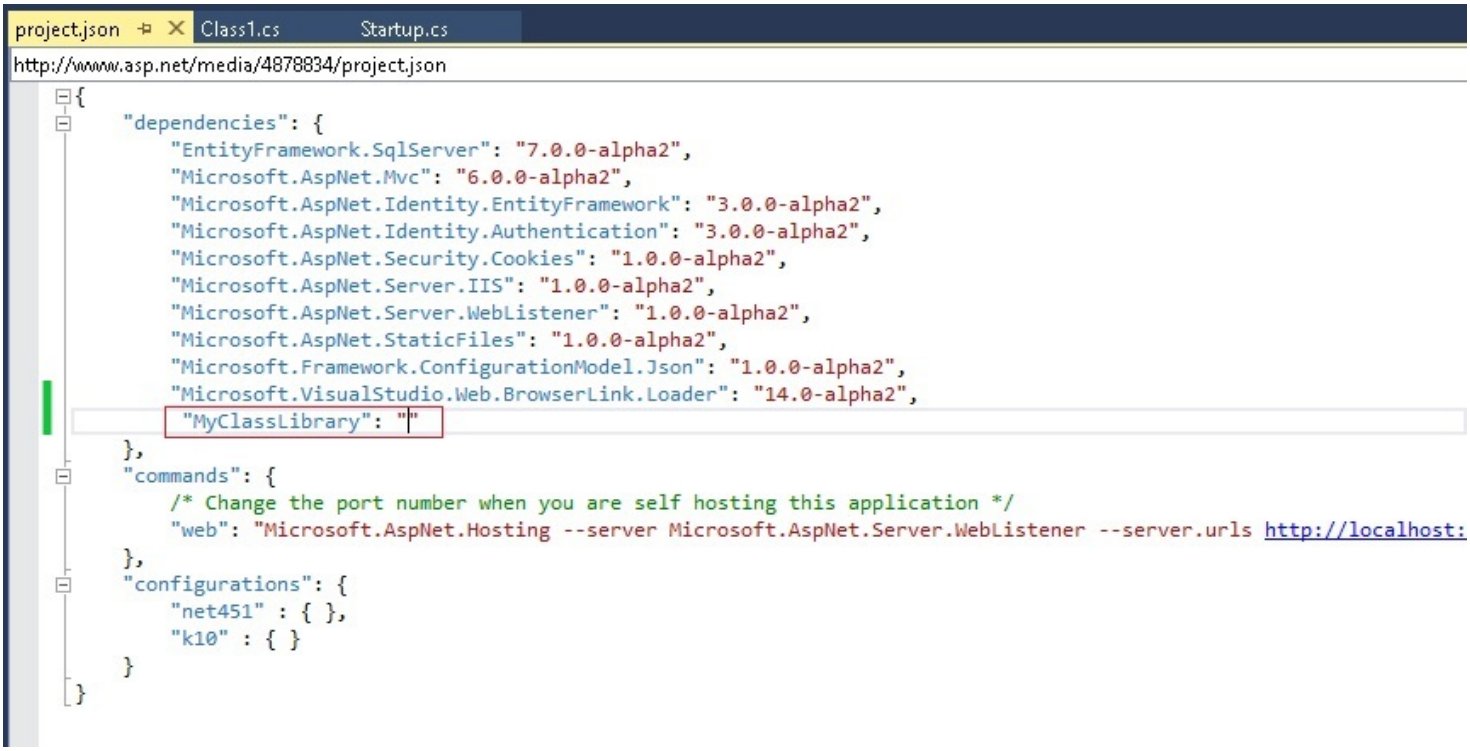


Now, since we have added this new project in our solution. Let's try adding a reference in our web project. So, in order to add the reference of this project, we need to list this in our "***project.json***" file as shown below.

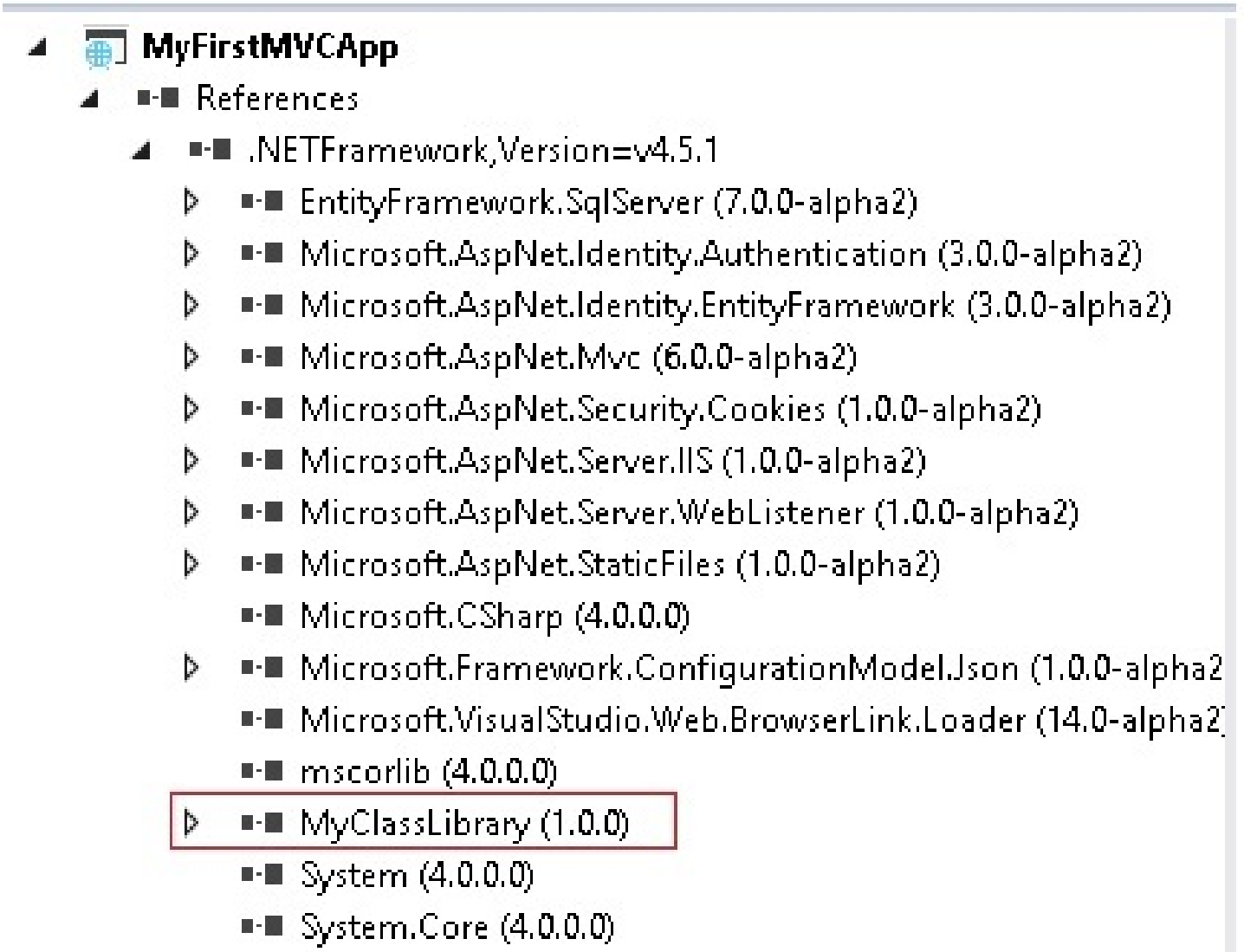


So, as you can see in the above screen shot, "***MyClassLibrary***" got listed in the intellisense, so that's nice pretty mature

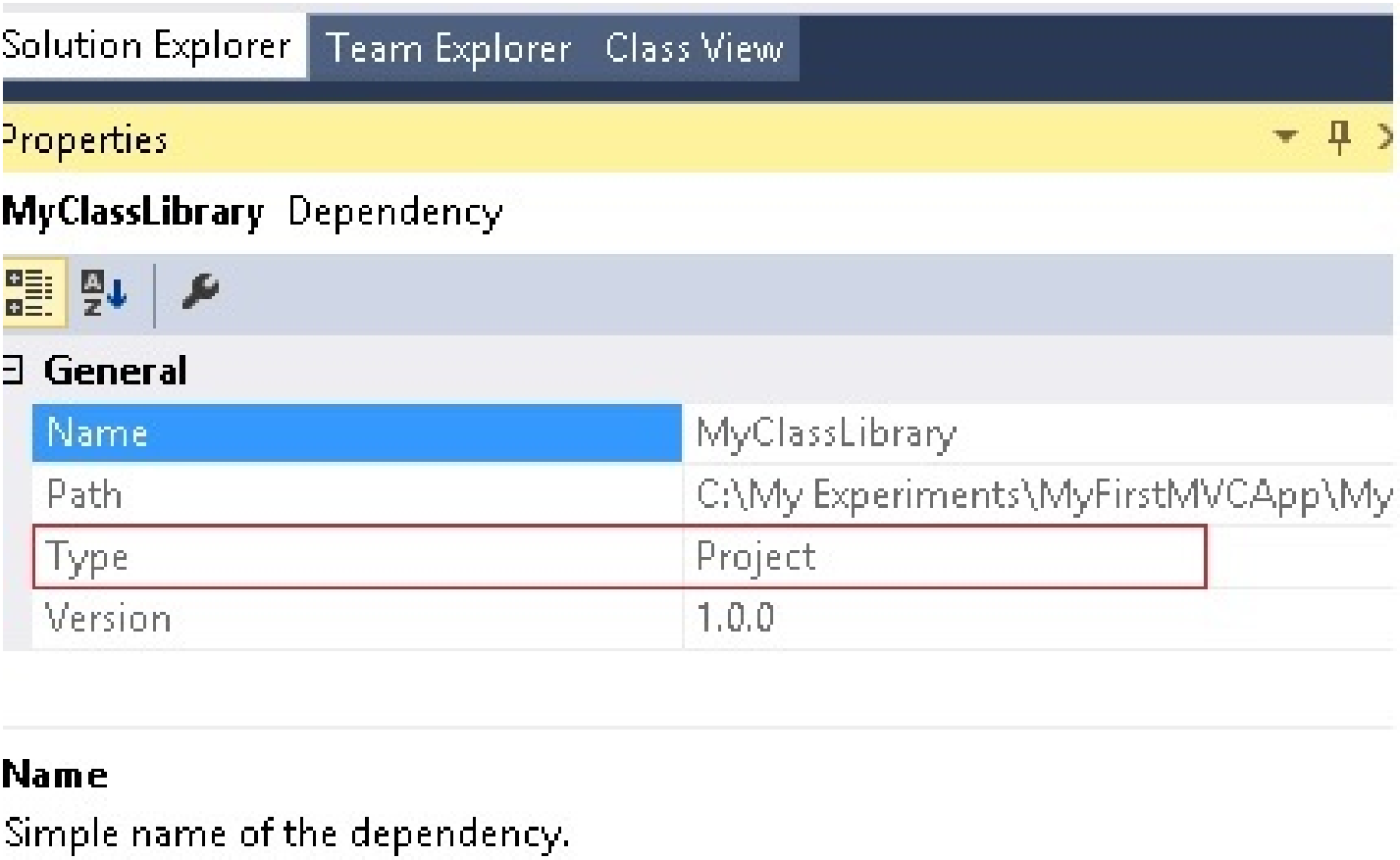
config system, no chance of mistake here.



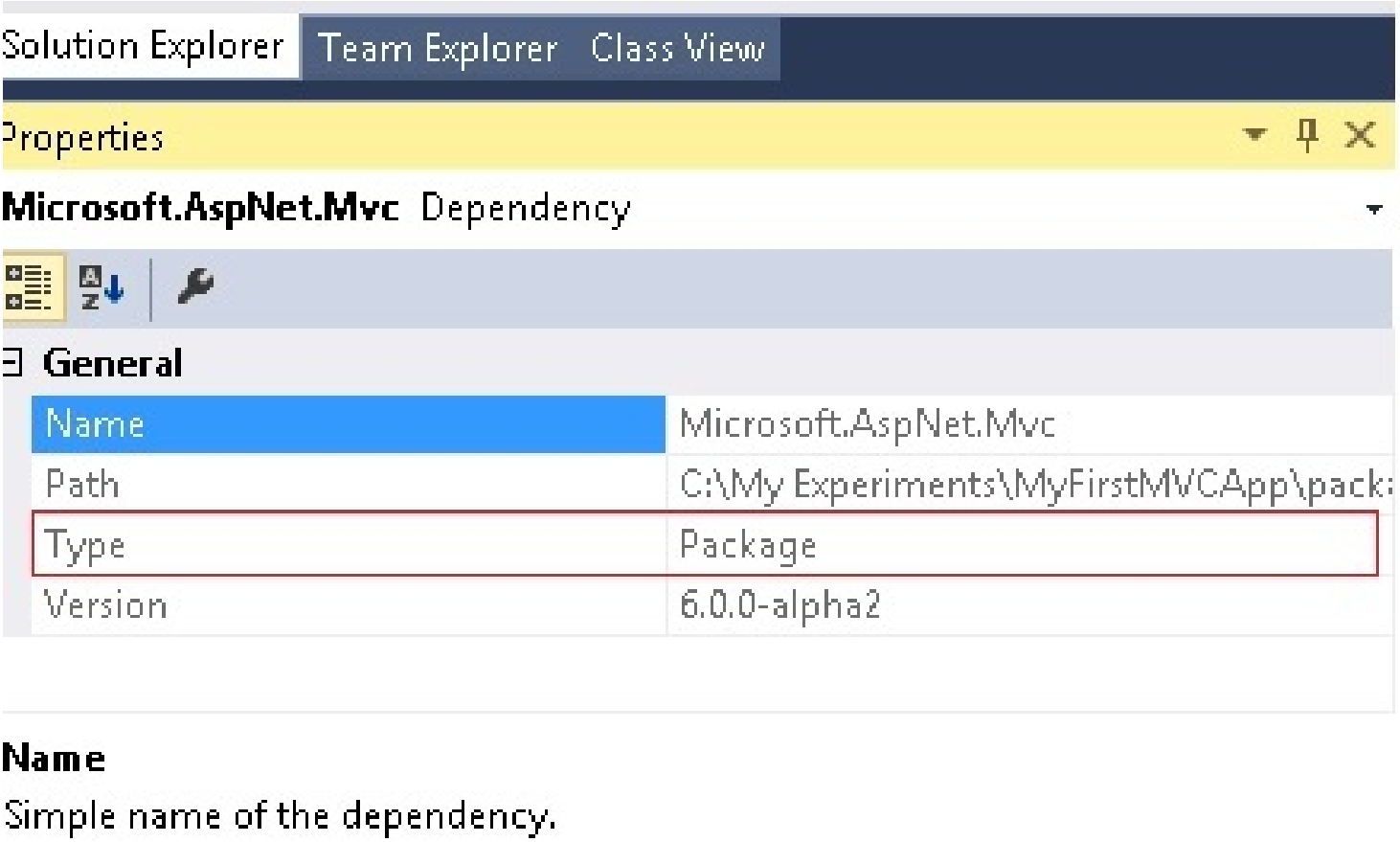
Now, since I saved this file, so I can verify the references as well.



Now, when you check the property of "***MyClassLibrary***", it will be of type "***Project***" as shown below.

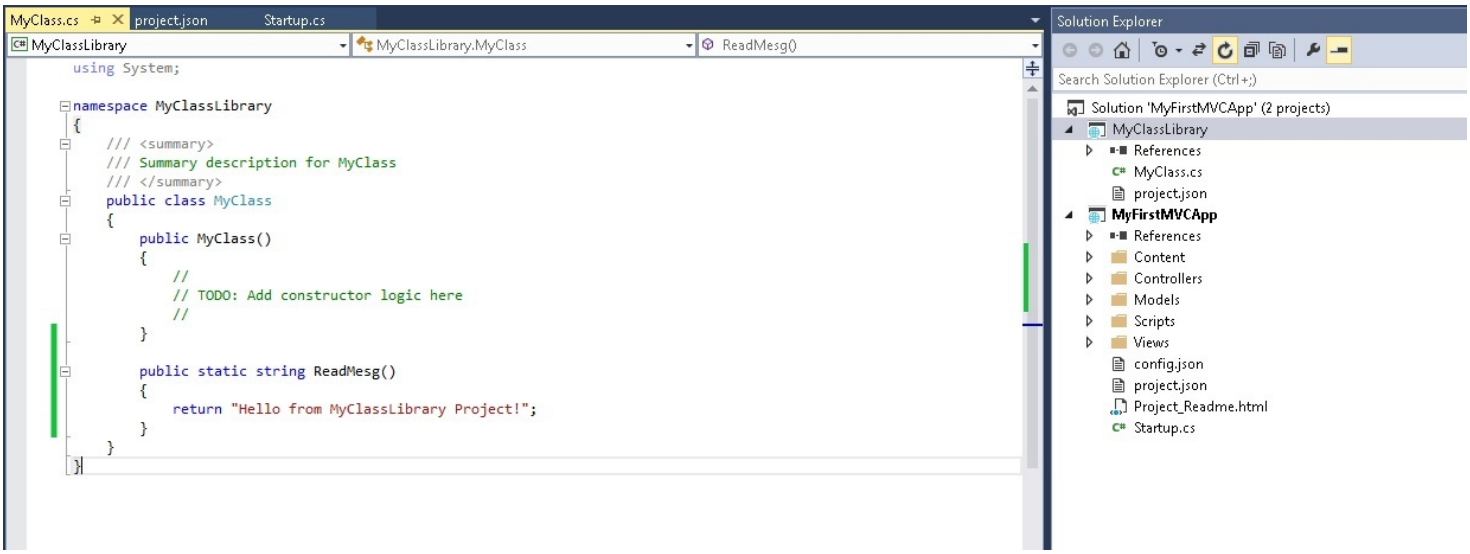


On the other hand others are of type "***Package***" as shown below.

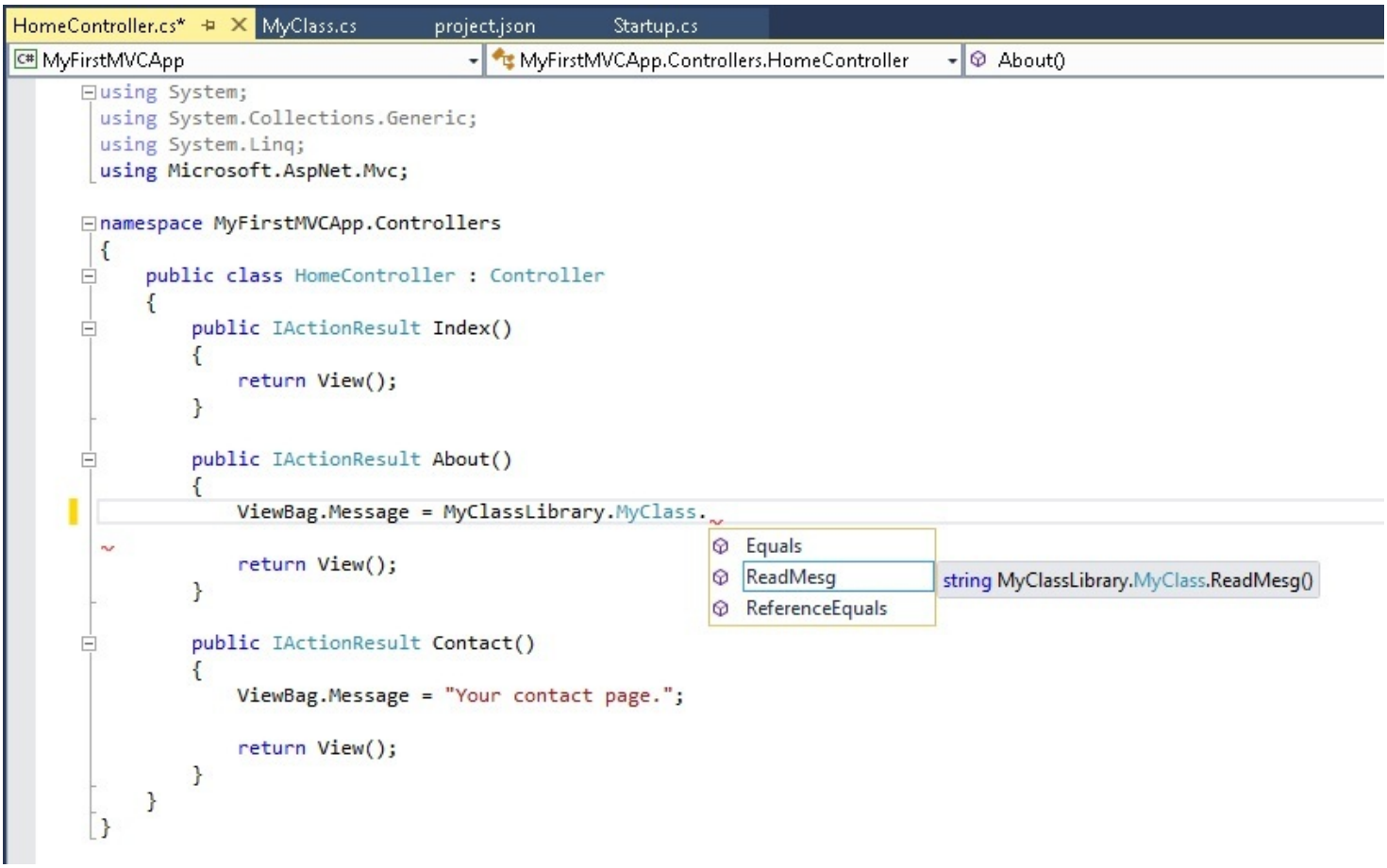


Since I have got the library project referenced in my project, let's try reading value from the class library project in one of our controllers. So, let's delete the default class provided in the class library project and create one new class with the name "***MyClass***"

with a static method as shown below.



Now, let's use this in our "***About Controller***".


HomeController.cs*
MyClass.cs
project.json
Startup.cs
MyFirstMVCApp
MyFirstMVCApp.Controllers.HomeController
About()
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.AspNet.Mvc;
namespace MyFirstMVCApp.Controllers
{
public class HomeController : Controller
{
public IActionResult Index()
{
return View();
}
public IActionResult About()
{
ViewBag.Message = MyClassLibrary.MyClass.
return View();
}
public IActionResult Contact()
{
ViewBag.Message = "Your contact page.";
return View();
}
}
}
Equals
ReadMesg
ReferenceEquals
string MyClassLibrary.MyClass.ReadMesg()

```
MyFirstMVCApp                    MyFirstMVCApp.Controllers.HomeController    About()

using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.AspNet.Mvc;

namespace MyFirstMVCApp.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }

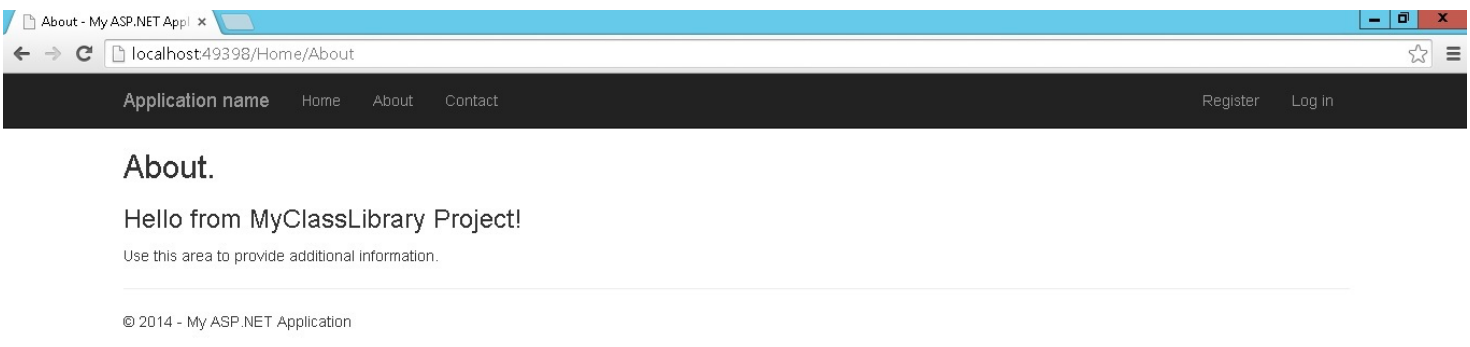
        public IActionResult About()
        {
            ViewBag.Message = MyClassLibrary.MyClass.ReadMesg();

            return View();
        }

        public IActionResult Contact()
        {
            ViewBag.Message = "Your contact page.";

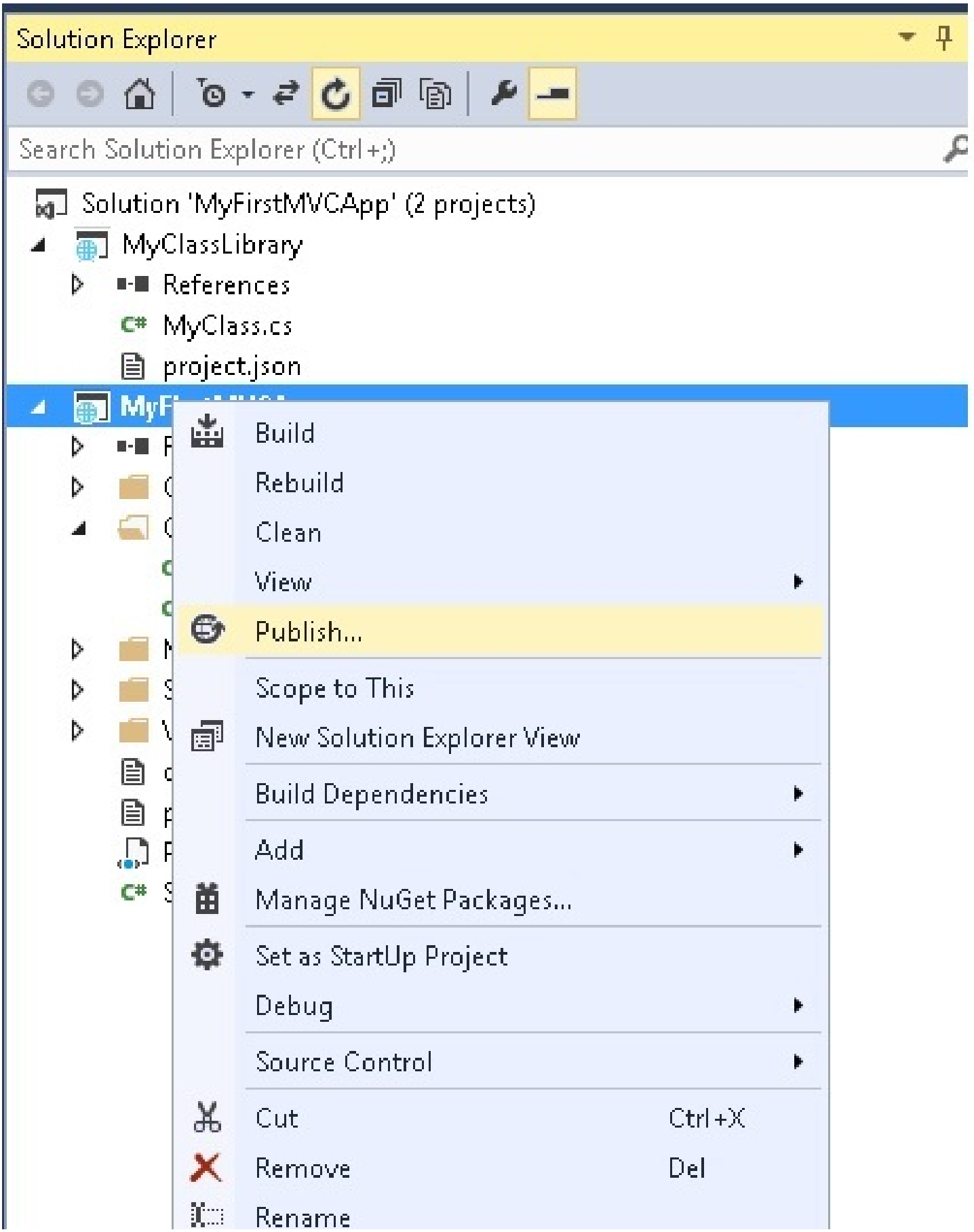
            return View();
        }
    }
}
```

So, now I saved this. So, when I refresh the browser without even building my solution, it will take effect. So, again behind the scenes, it's "***Roslyn***" magic.

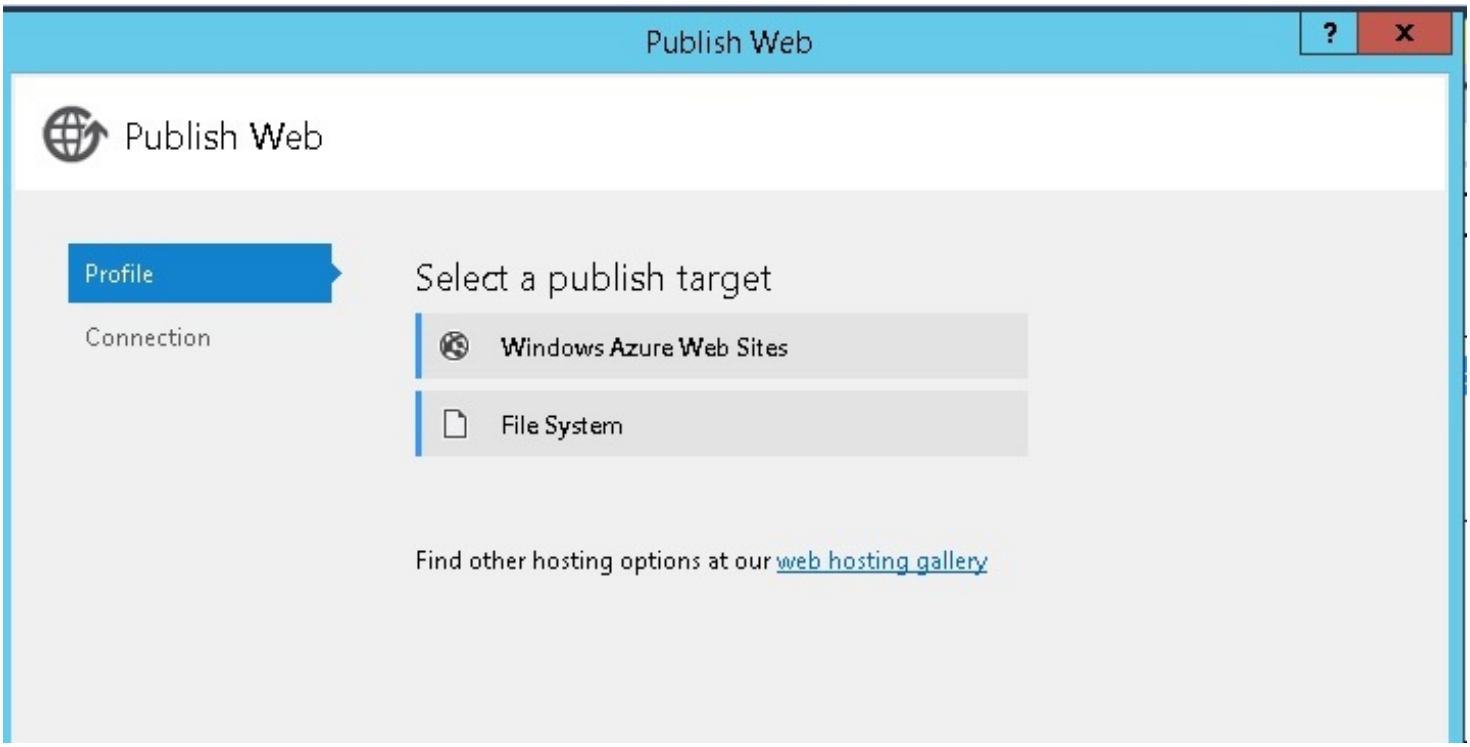


Deployment to Azure:-

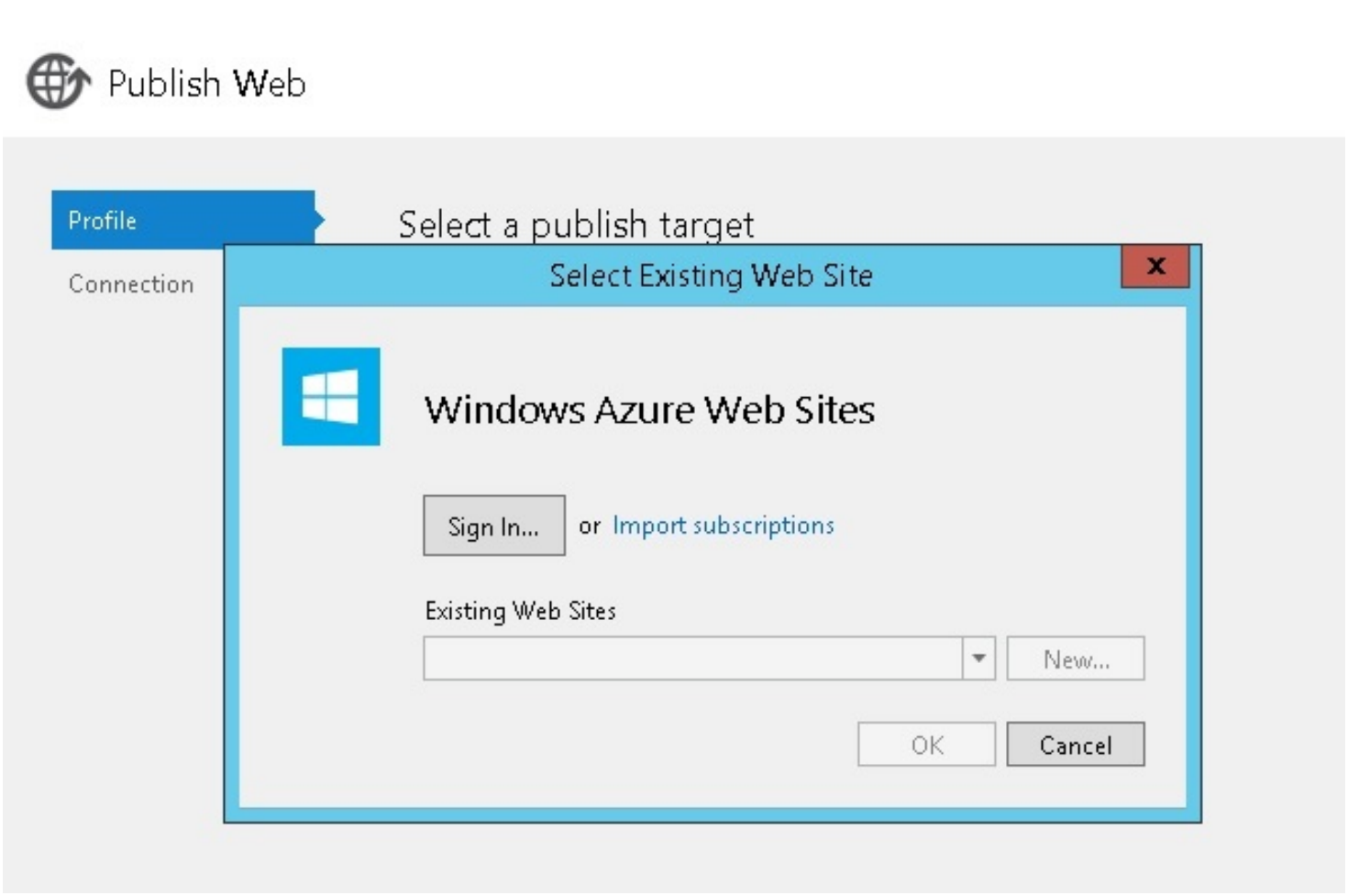
Now, in this section, I am going to demonstrate little different way to host the site directly in Azure. So, here, currently I am on my VM (Virtual Machine). I have running app on this machine and from this machine, I'll publish my site on cloud. So, to do the same, right-click on the web project and click on publish option as show below.


Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'MyFirstMVCApp' (2 projects)
MyClassLibrary
References
MyClass.cs
project.json
Build
Rebuild
Clean
View
Publish...
Scope to This
New Solution Explorer View
Build Dependencies
Add
Manage NuGet Packages...
Set as StartUp Project
Debug
Source Control
Cut Ctrl+X
Remove Del
Rename

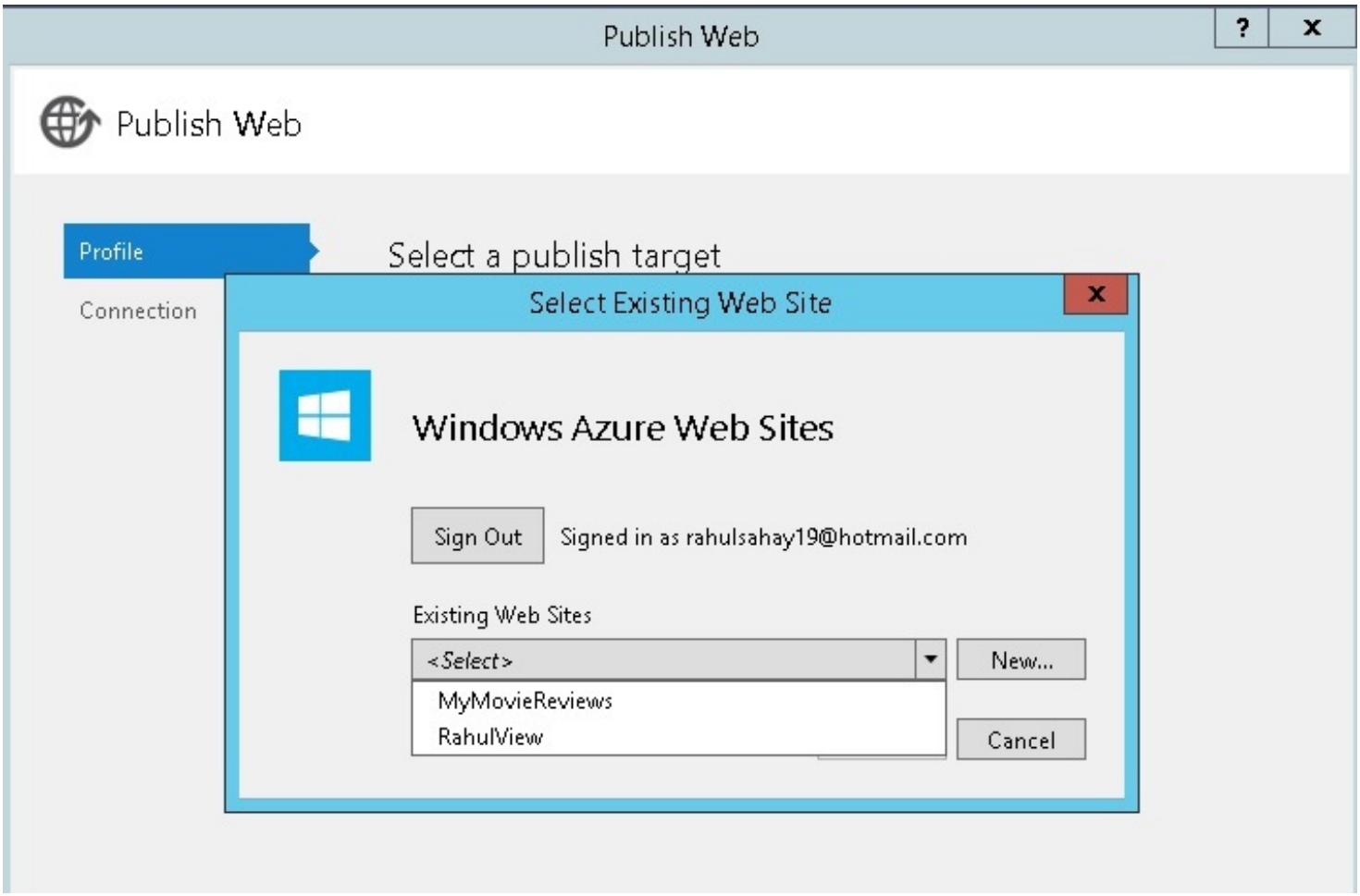
Then, from the publish wizard, just click "***Azure Web Sites***".



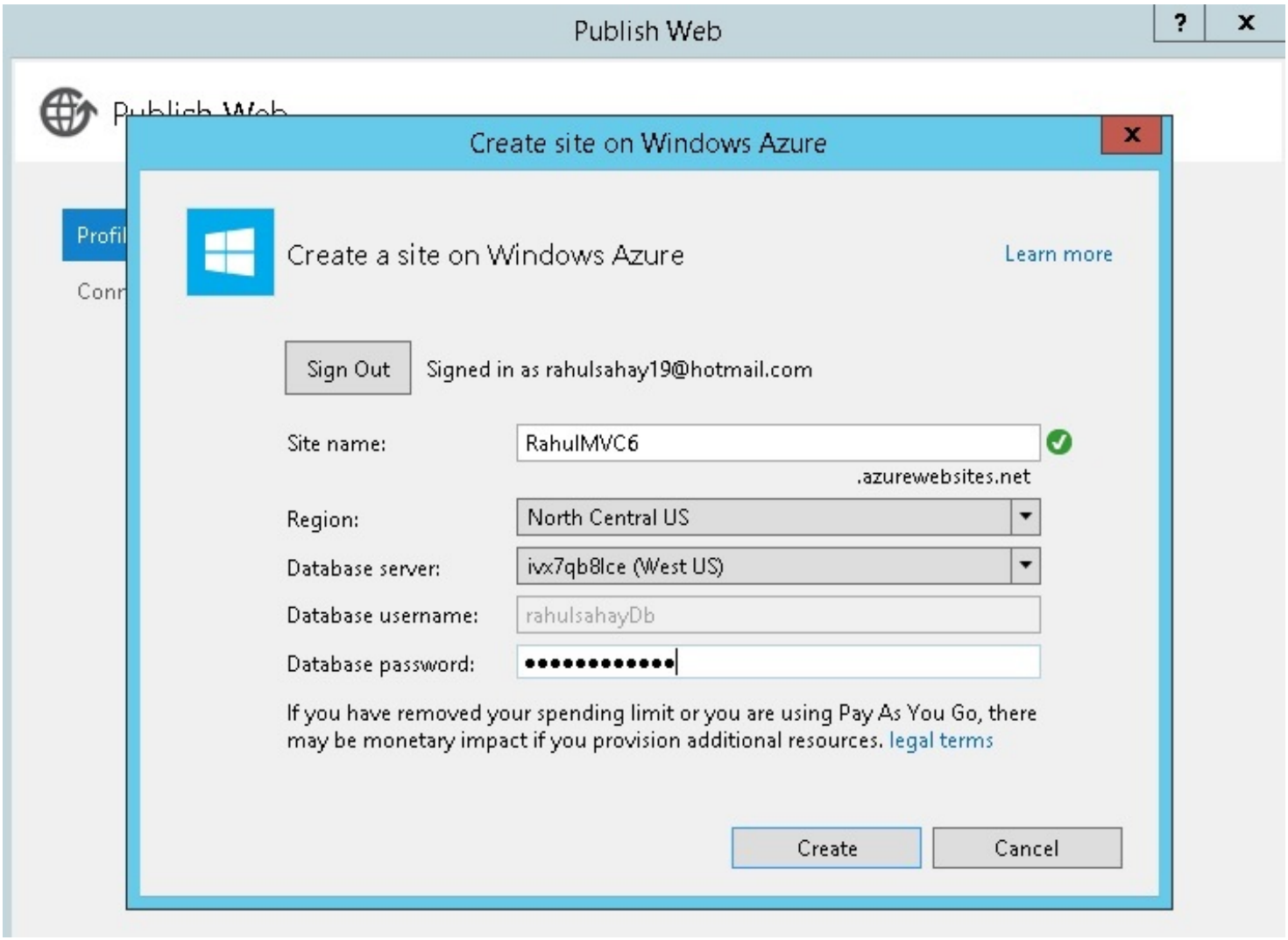
Then, provide your credentials and login



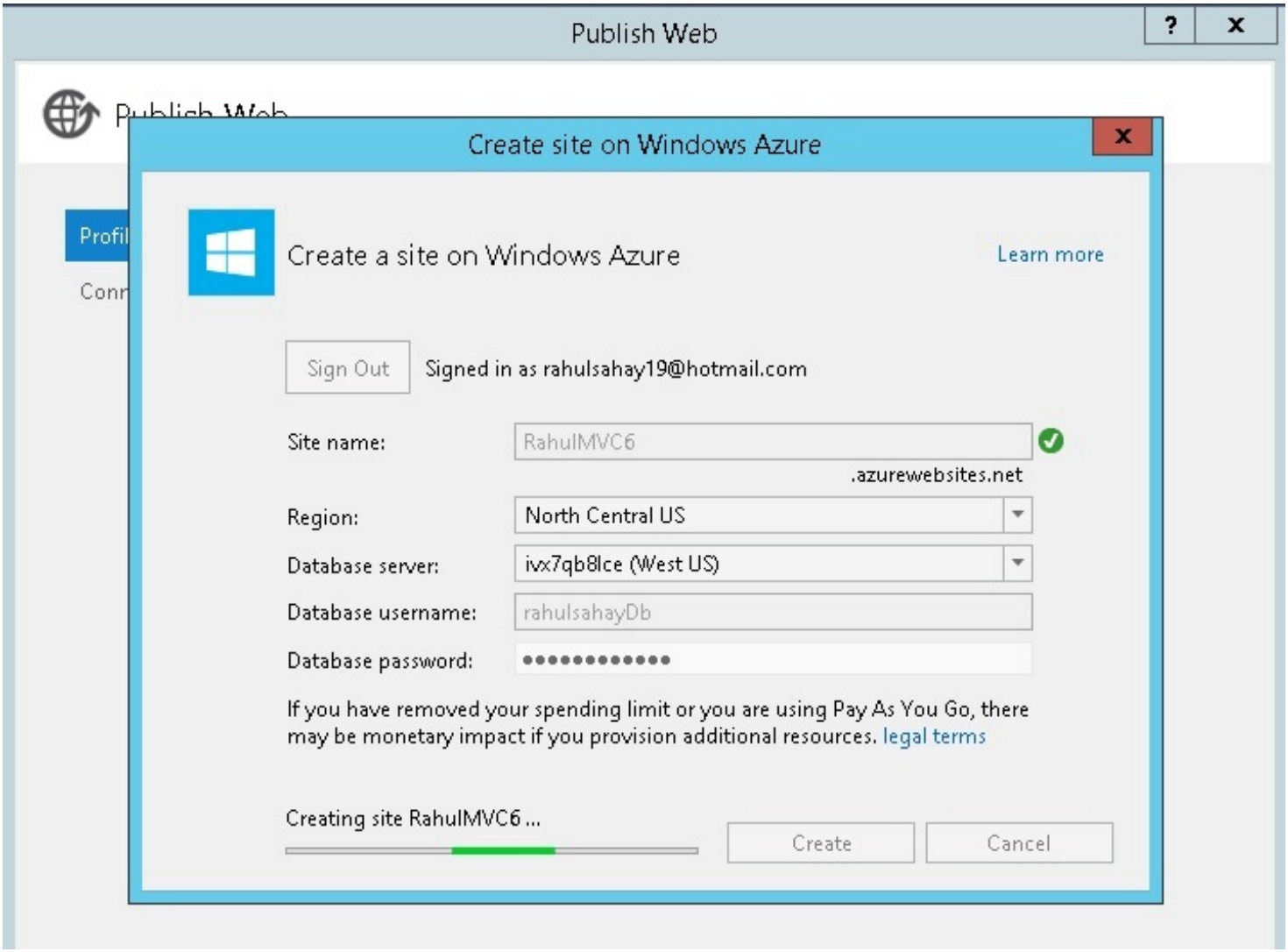
Now, one point to note here, I am going to create new Azure website right from here. Earlier, during movie review site deployment, I have already created one site for that and then I deployed the files there. So, this is another technique to deploy your website on Azure.



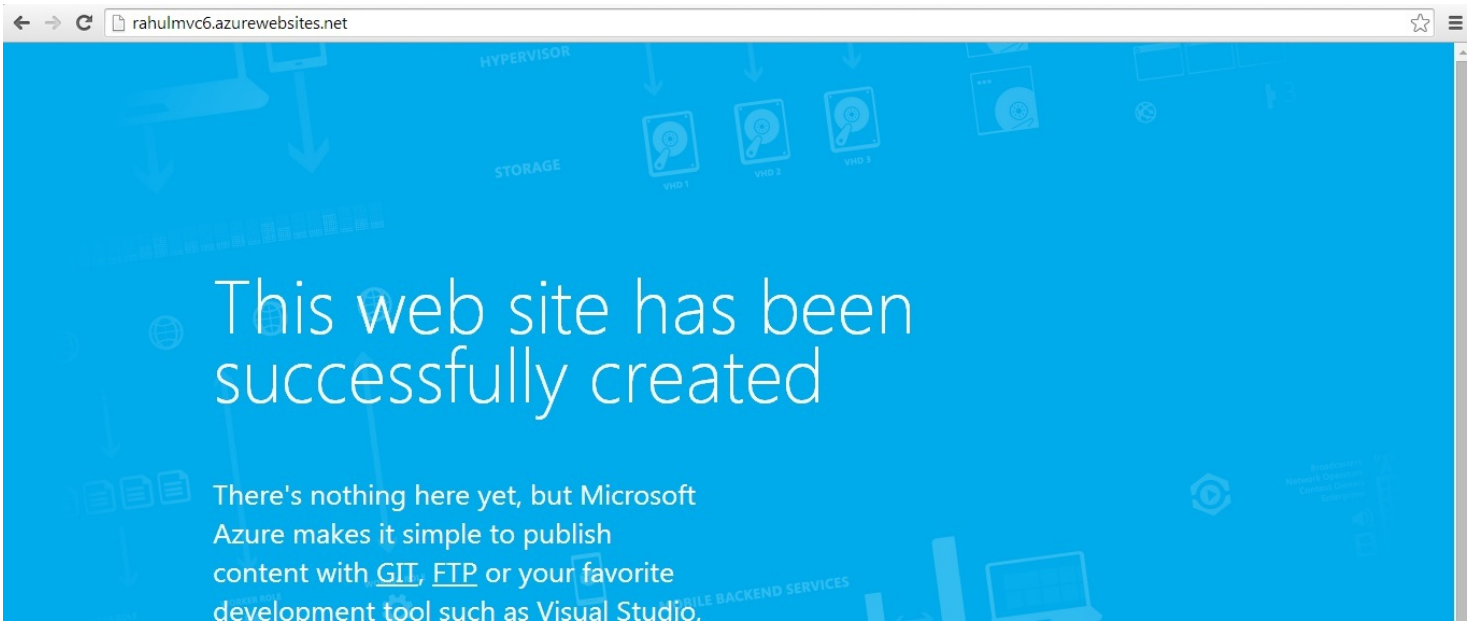
So, as you can see in the dropdown, it is showing me the sites which I already hosted on my Azure. So, basically this option is for that case when you want to deploy some changes on the existing web site. But, here I'll go with the "***New***" option as I have to create a new website for this new MVC 6 application.



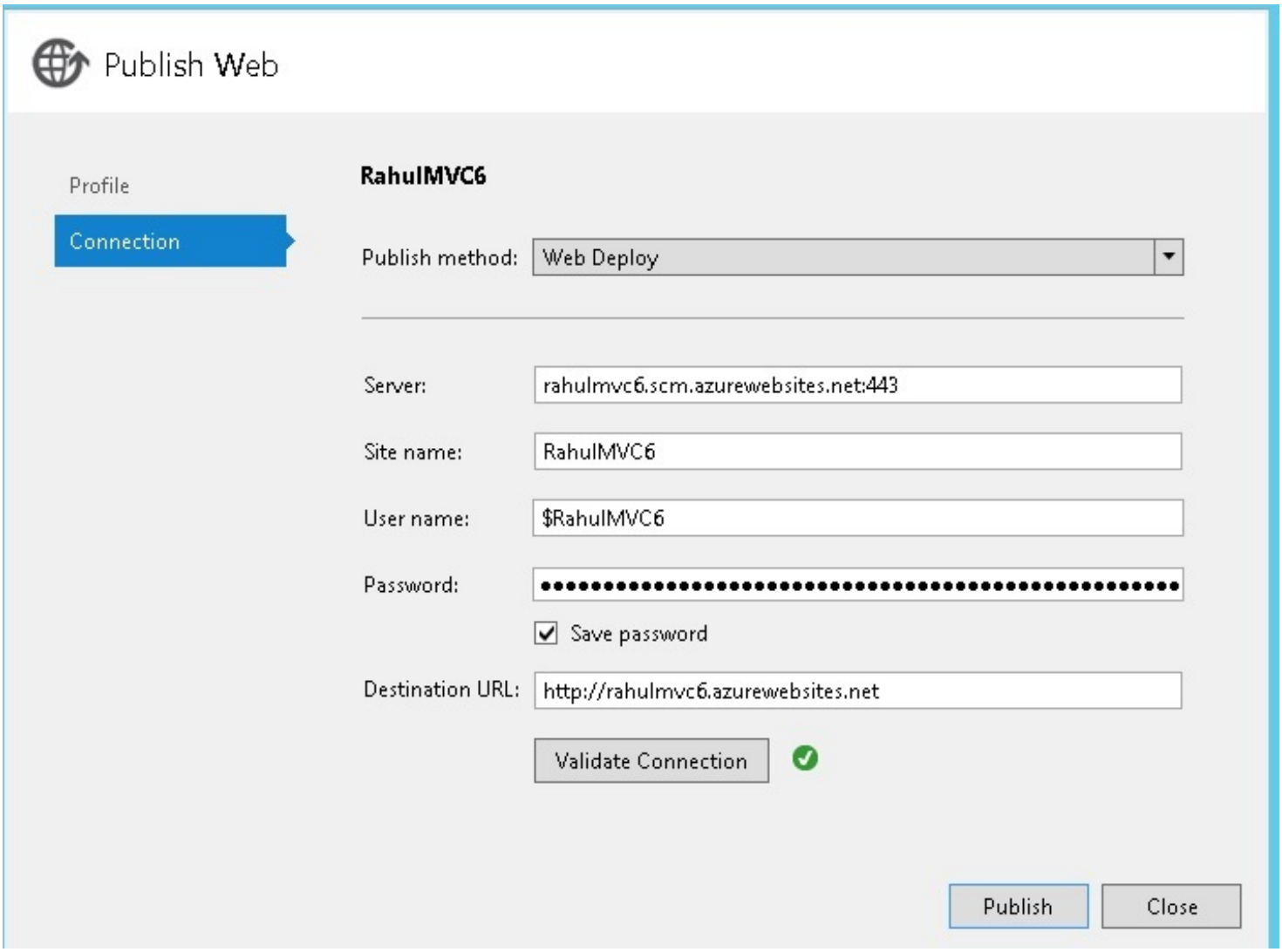
So, in the above screen shot I have already provided the required details. Now, I click on create button and then it will take all the required dependencies from the site; package them together and deploy the same on Azure website as shown below.



Then, upon successful creation of site, if you navigate the same in the browser, then it will show you the default page of the Azure as shown below.



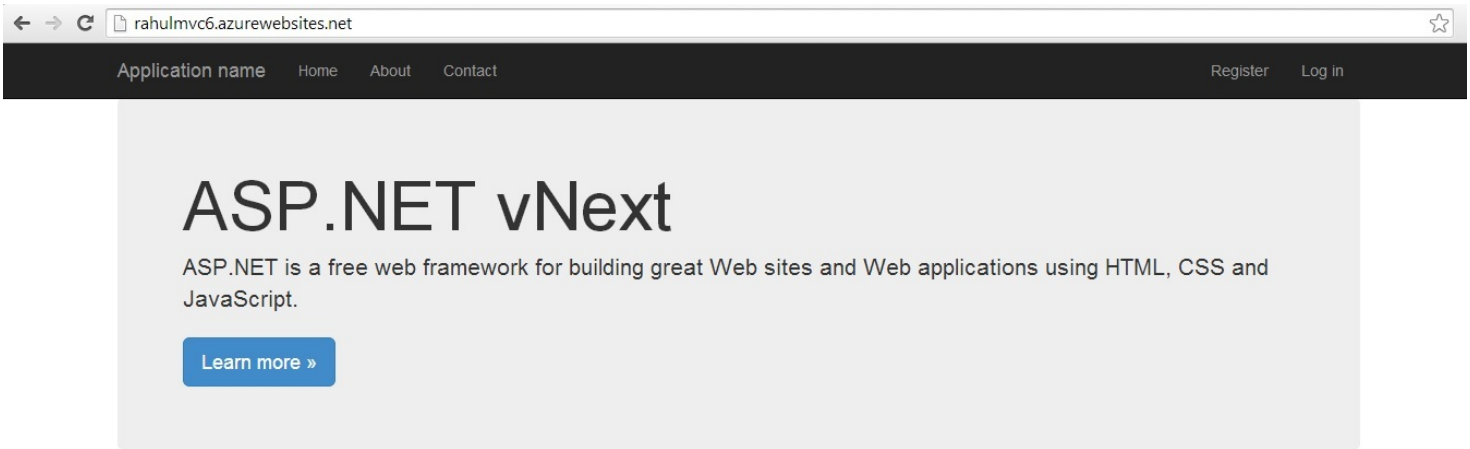
Then from connection tab, there will be "***Publish***" button which will bundle the files and push the same on the Destination URL as shown below.



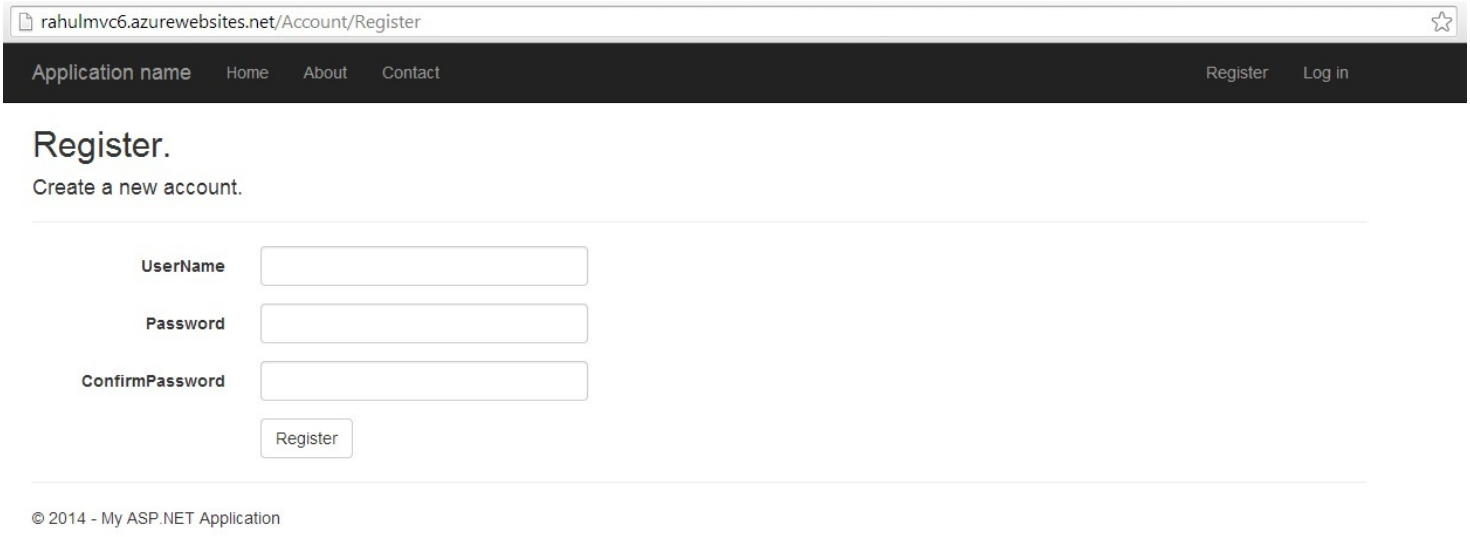
After successful publish, it will flag the message in the activity window as shown below

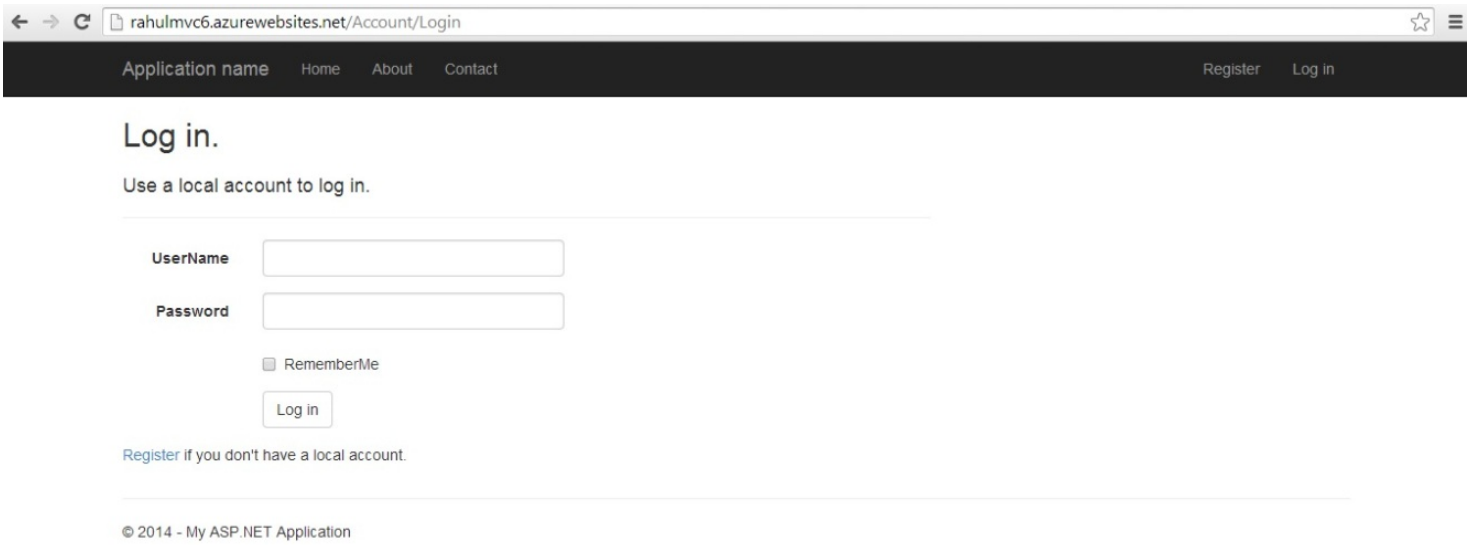


So, now, when I refresh the page in the browser, it will show my recently developed MVC 6 website on Azure.



So, this is a very basic site with all the stuffs in there like **Register/Login** etc. But, it's a good starting point to get started.

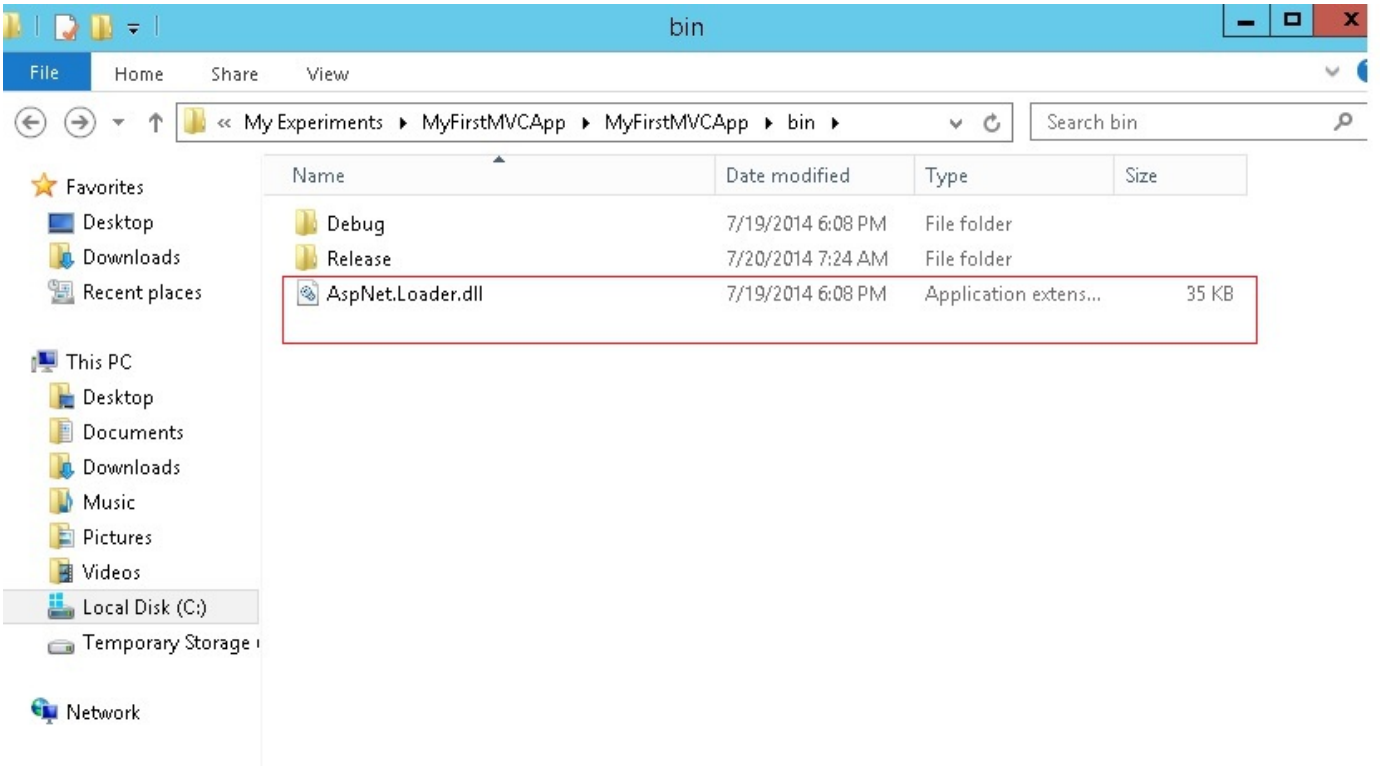




Bin:-

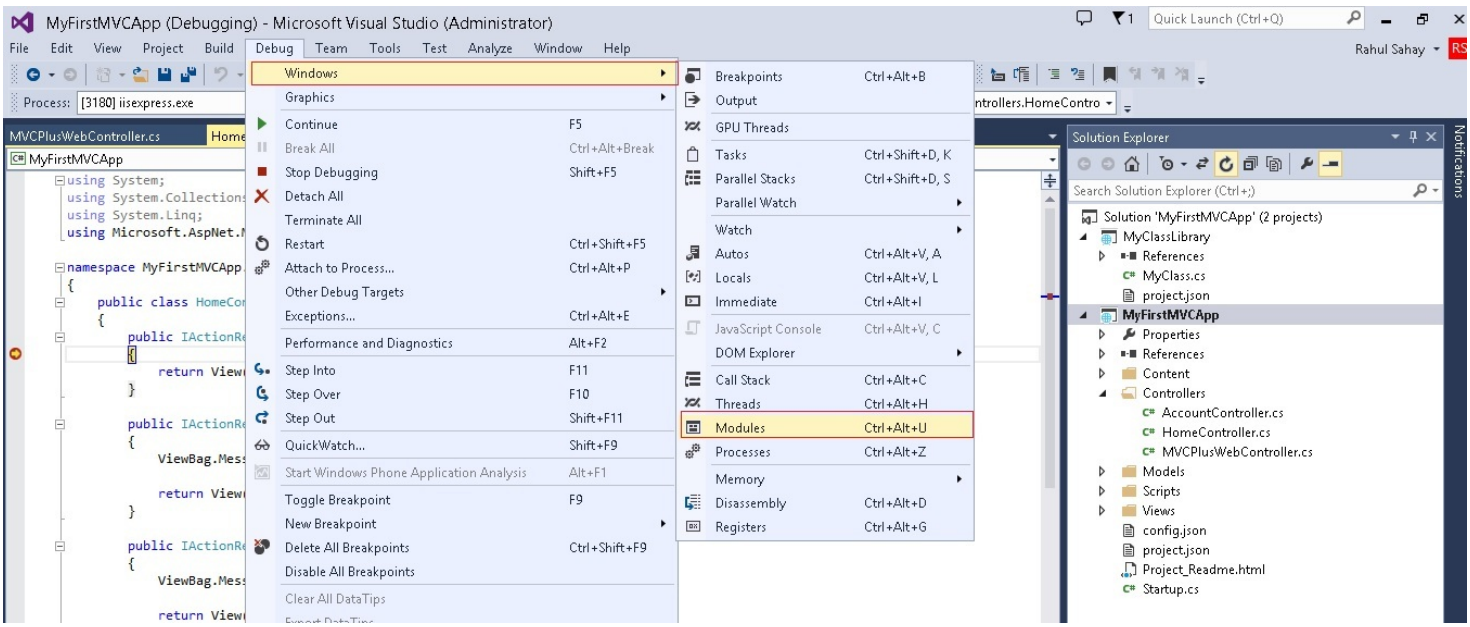
Now, in this section, we are going to do a bit deep dive. So, in this section, we'll

first explore that our project file specially bin directory.

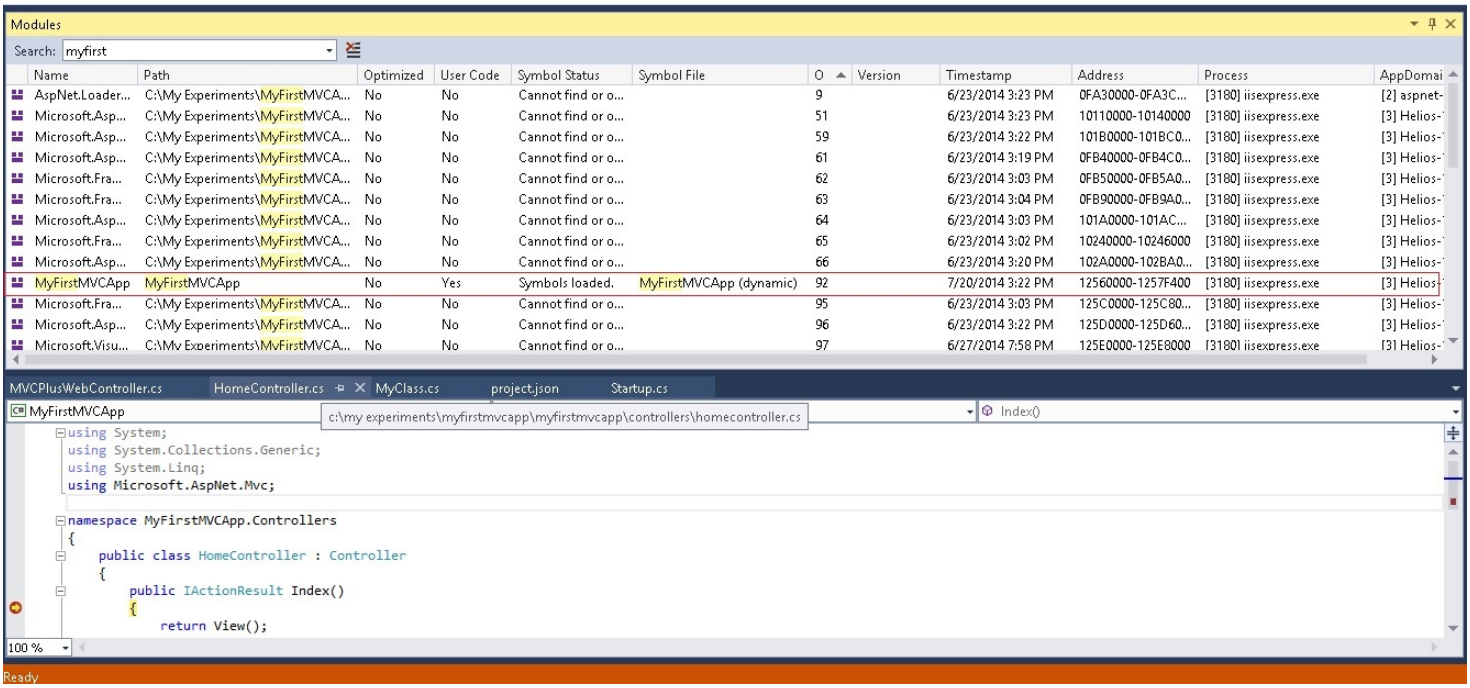


Here, you will notice one new thing, "***AspNetLoader.dll***". Now, this dll is pretty dynamic, it will take care of your code changes without build. So, every code change or "***cs file***" change is a complete refresh of process. Now, one more point to notice here, that you won't

find any dll either in "***Debug/Release***" folder neither "***.pdb***" file. Now, "***.pdb***" file is the one which enables user to debug the code. Then, how can developer debug the application. Well, to answer this question, it's dynamic in memory. So, when you see the "***Modules***" section under "***Debug Window***", you will find this critical piece of information.



Now, this will open a modules window as shown below in the screen shot.

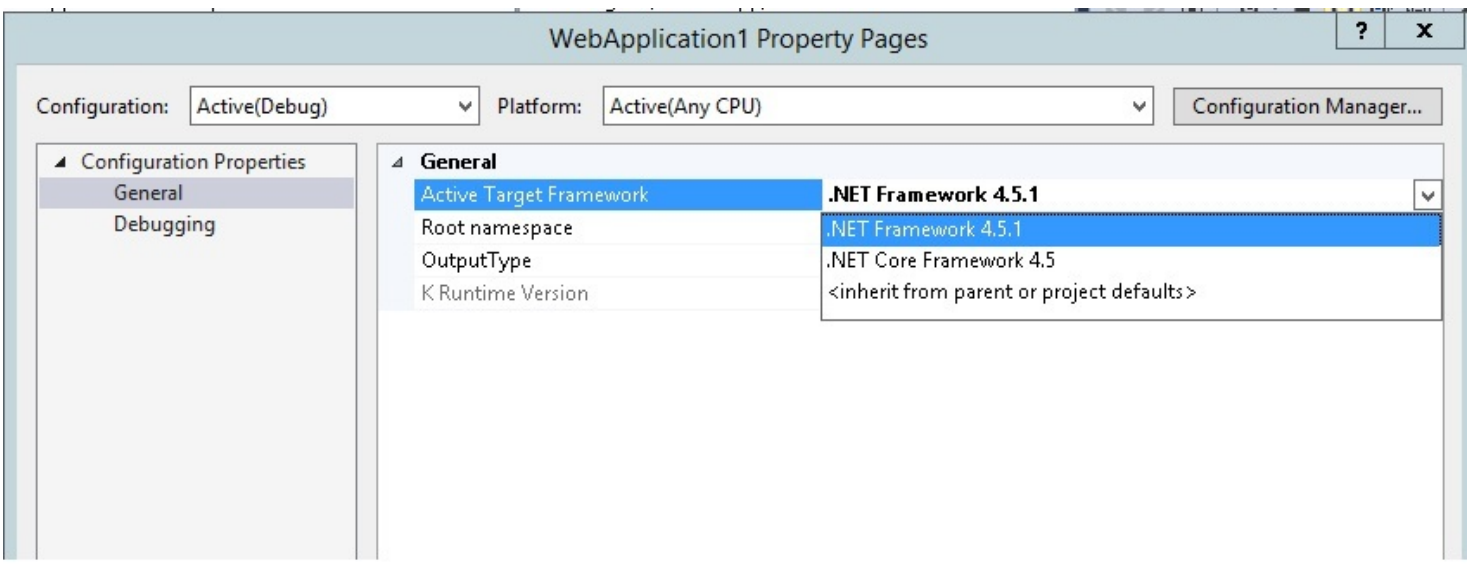


So, if you watch the above highlighted section closely, you will notice that it is loading the symbols to debug the file dynamically; that is why no path is there. Now, dynamic here means in memory. This also means "***.pdb***" is dynamic.

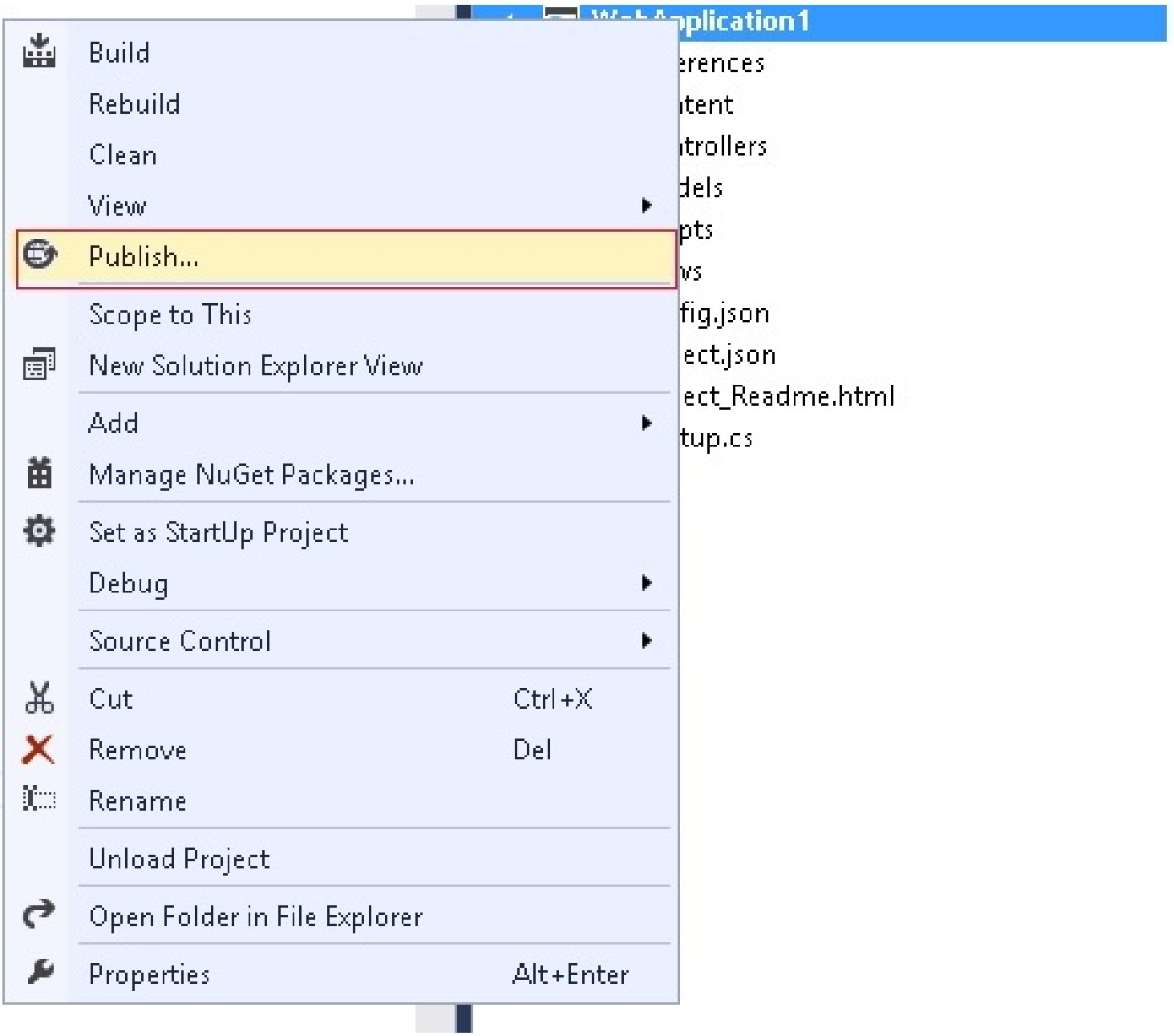
Self Hosting:-

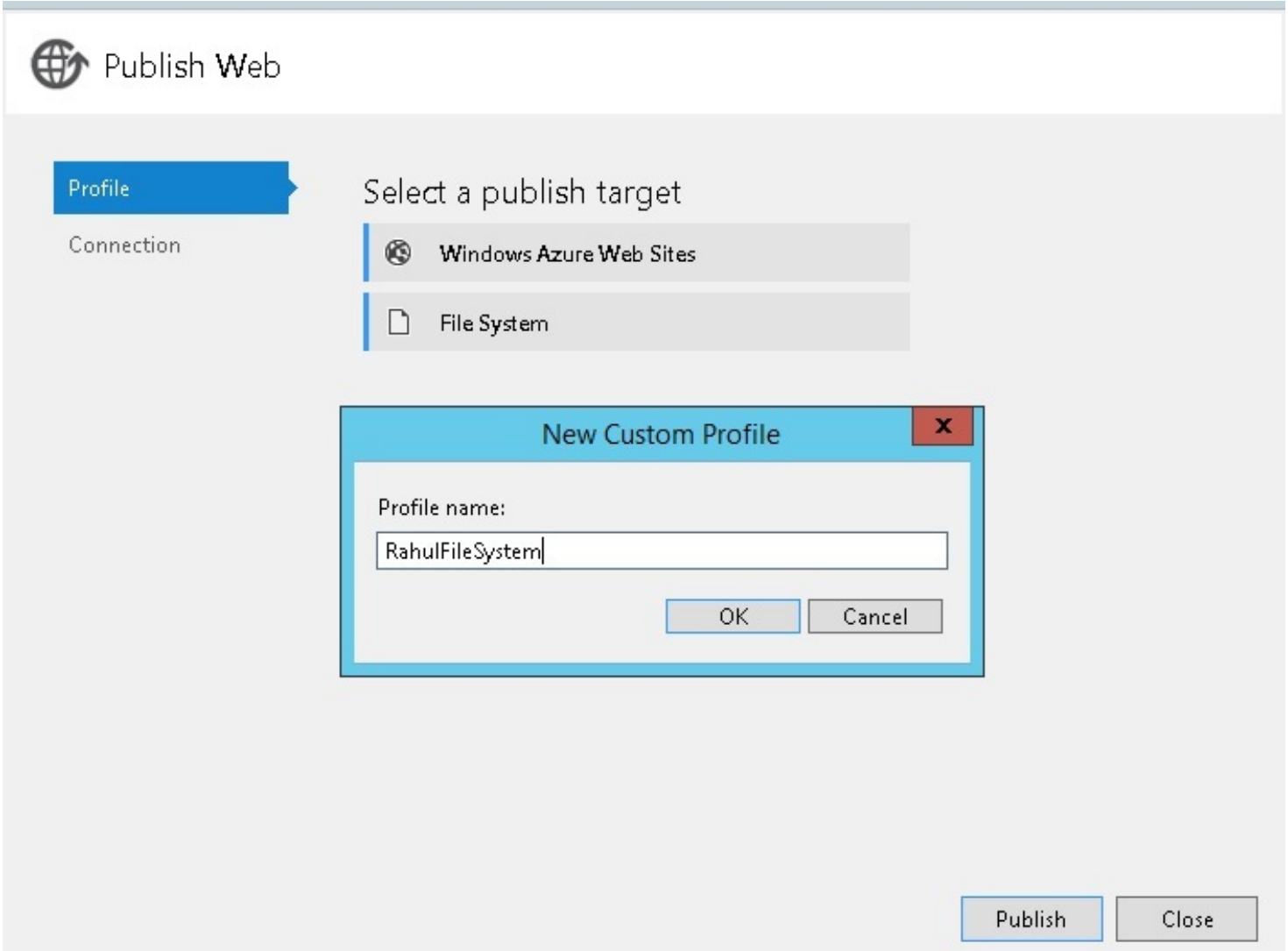
Now, in order to self-host the files, you need to follow couple of steps. These are mentioned below.

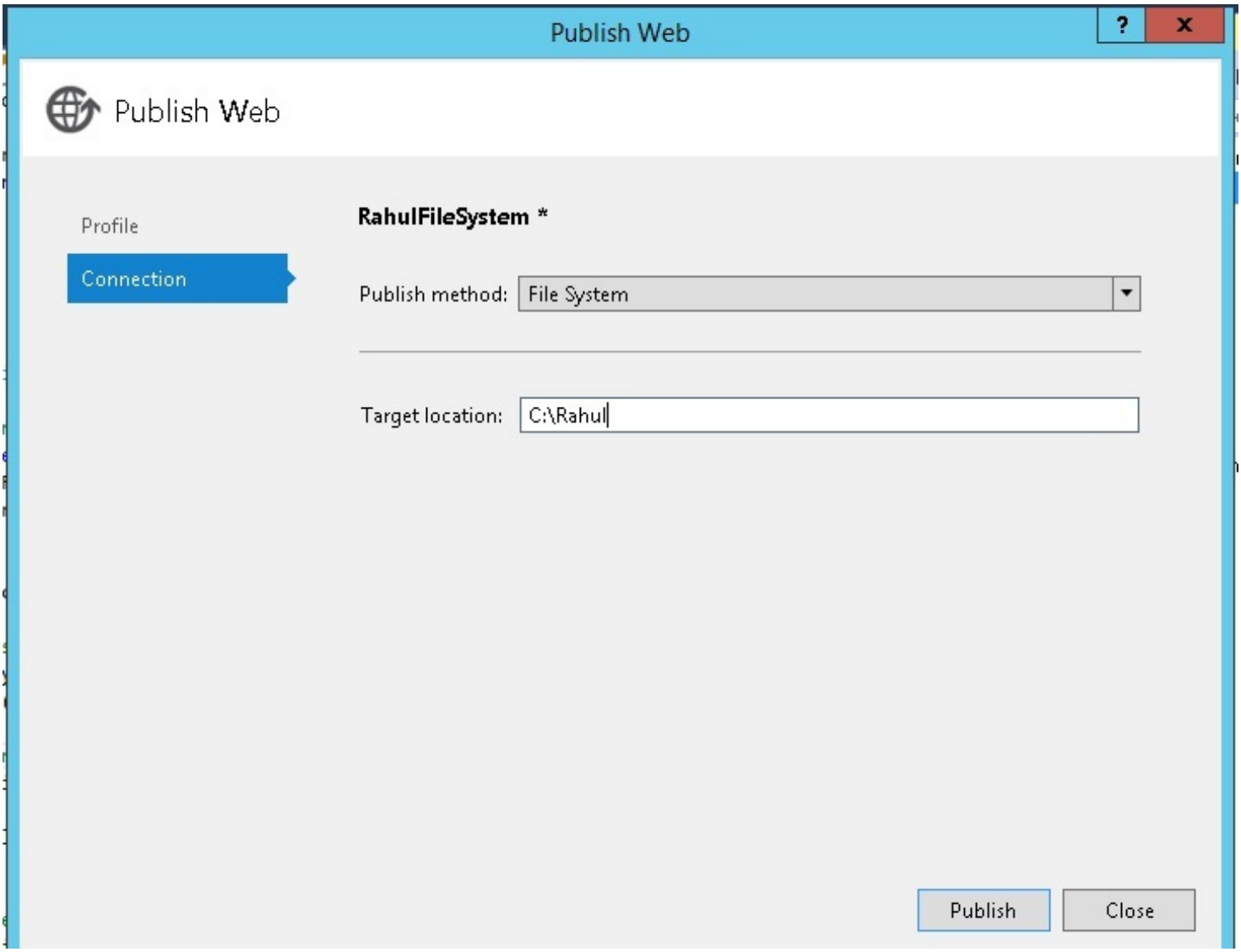
You need to change the "***Active Target Framework***" for the project to "***4.5***", which is cloud optimized version. Currently, it's running on the desktop version which is "***4.5.1***".



So, now I have set the Framework to "***4.5***". Now, I'll right click the project and publish the same on the "***File System***" as shown below.



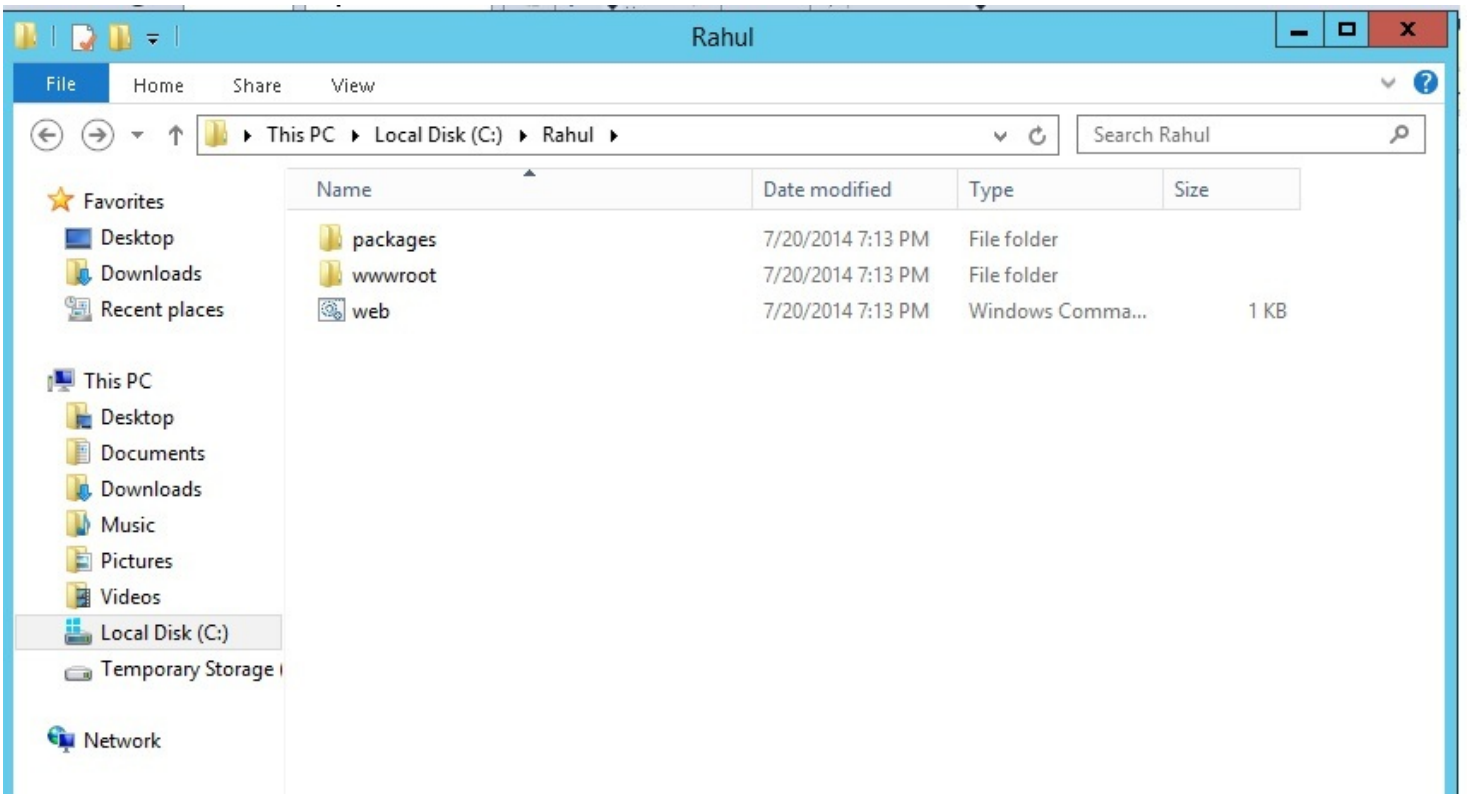

Publish Web
Profile
Connection
Select a publish target
Windows Azure Web Sites
File System
New Custom Profile
Profile name:
RahulFileSystem
OK
Cancel
Publish
Close



Then, upon clicking publish button it will publish the files at the desired location as shown below.

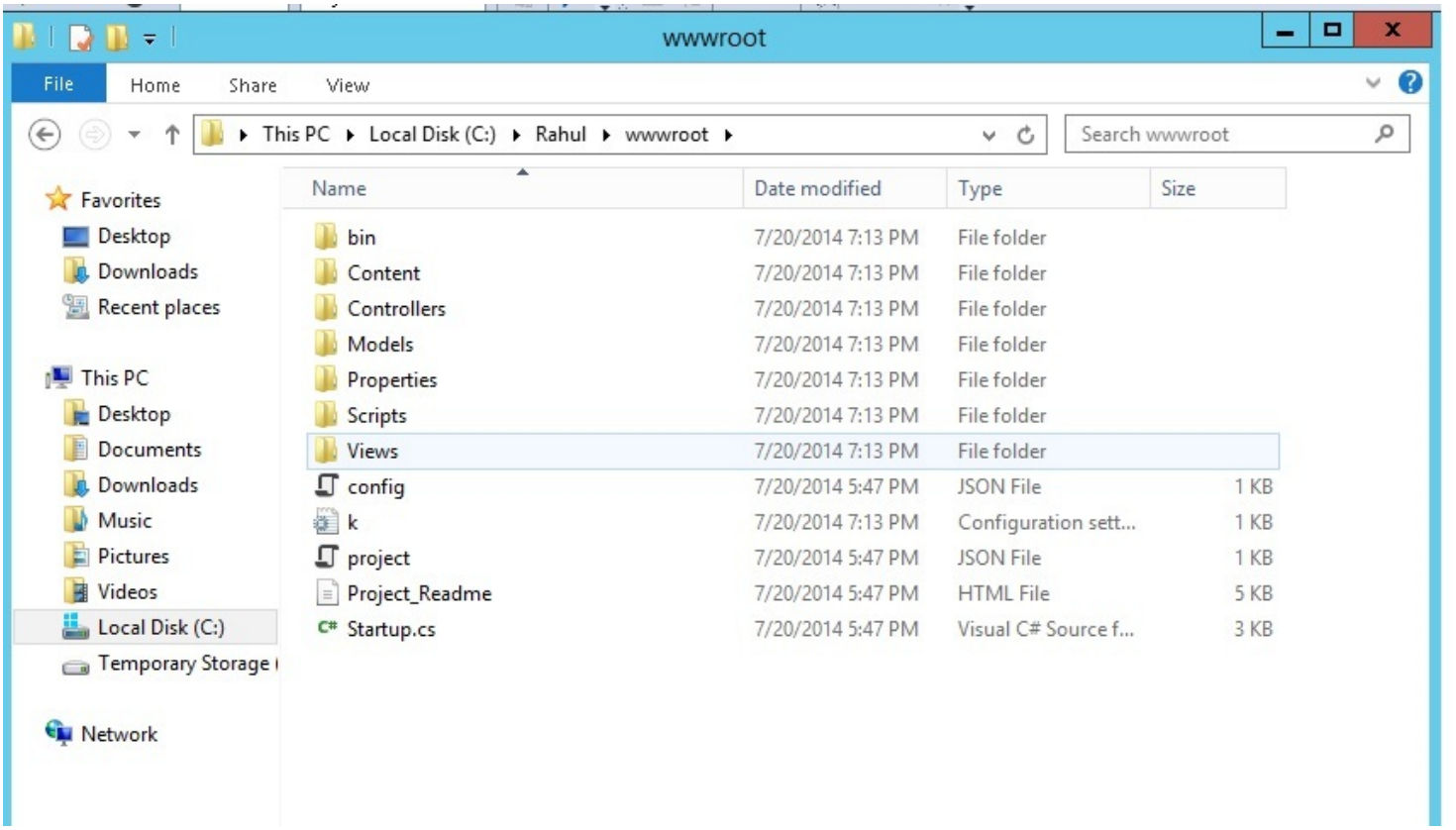


Now, when you open the folder, you will see all executables with a bunch of files.

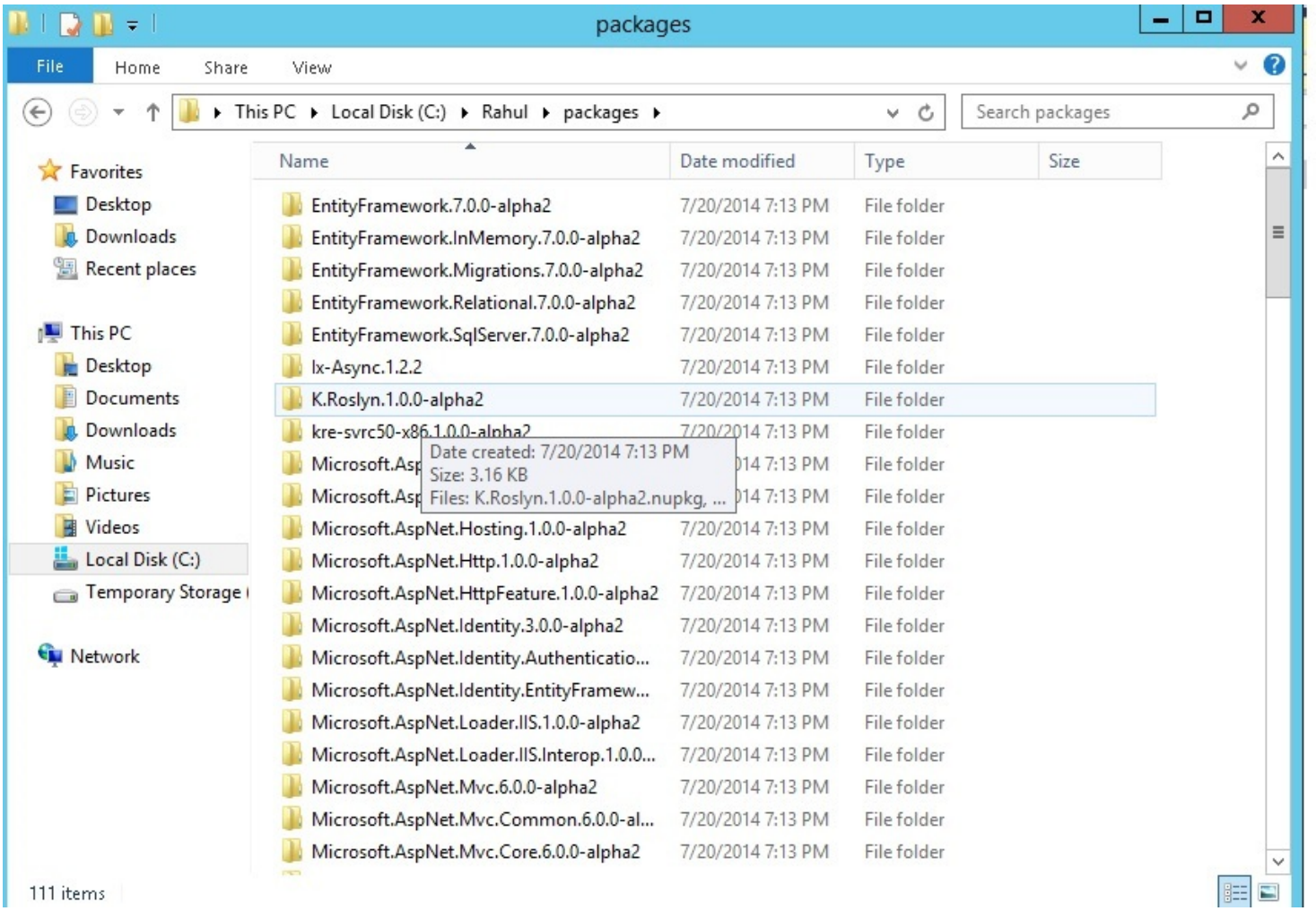


So, this web is going to be executed by the command line and if you see inside

the "***wwwroot***", it will have all the files required to get deployed on the localhost as shown below.



Similarly, it has all the dependent packages required to run the app.

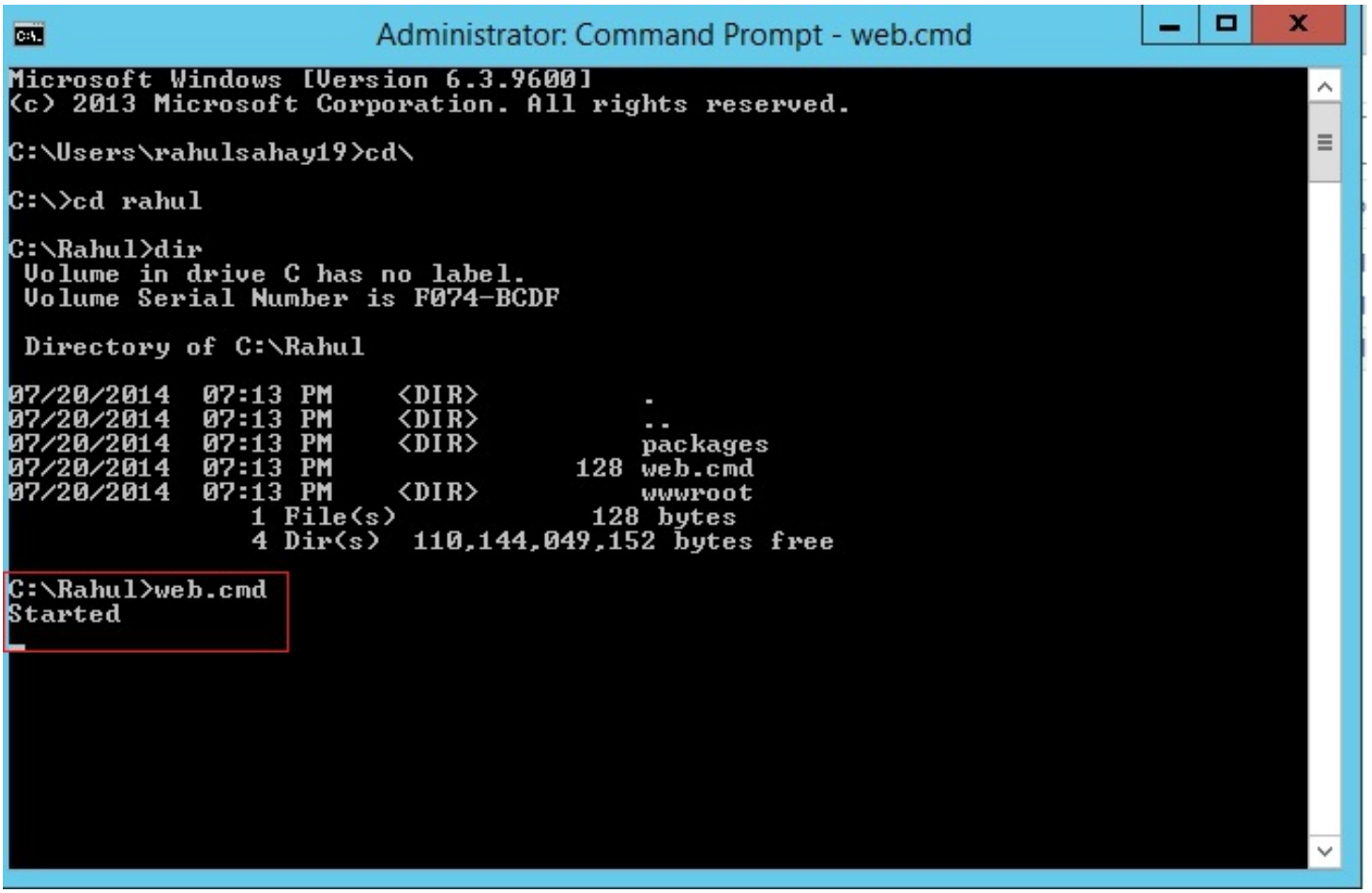


Now, if you open "***web.cmd***" in a notepad, it says go into the packages folder and run the "***klr.exe***", which is nothing but cloud optimized run time, you can think of this as the minified version of CLR, which is only project specific.

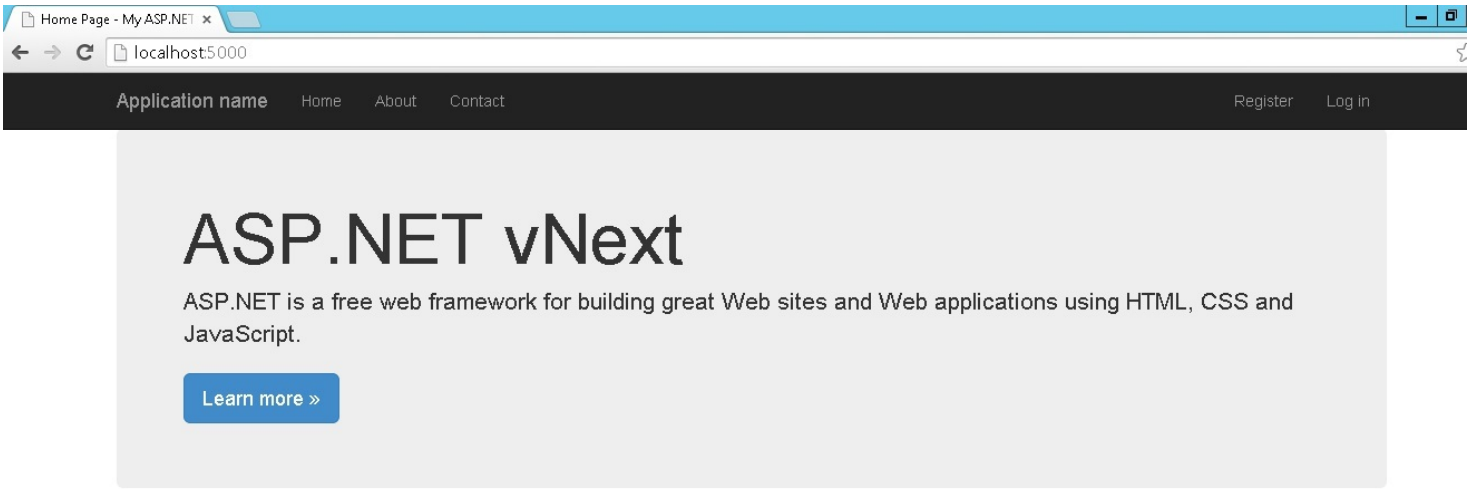


Now, this application is fully self-contained application which means you can take these files in the pen drive and publish the same from the pen drive. It doesn't have any dependency on CLR or any other Microsoft environment related thing. So, you can run the same from Linux box or MAC machine, doesn't matter anymore.

So, the next thing which I do here go in the "***cmd***" prompt, navigate to the location and then type "***web.cmd***", then after it will flag the message "***started***" as shown below.



Then, I'll navigate the self-hosted which is mentioned in project.json file ***http://localhost:5000/*** which will produce me the below output.



Summary:-

So, in this section, we have seen completely new kind of MVC or I would say pretty mature MVC which is going towards cross platform support. We have also covered vNext features, and also explored some of the exciting features of CLOUD (Azure). Like we have seen how to setup Virtual Machine with Visual Studio. Then, we have

created new vNext project. We discussed various aspects of this new vNext template in detail. We have not only hosted the same on Azure, rather we have also made KLR version of file system which can be hosted on any platform. I hope you have enjoyed the journey towards learning MVC.

INDEX

$

$ · 178, 179, 186, 187, 238

.

.js · 163, 169, 178
.NET Framework · 97

@

@ · 57, 59, 60, 63, 64, 65, 67, 70, 72, 73, 75, 80, 92, 94,
98, 102, 106, 117, 120, 137, 140, 143,

144, 149,
161, 169, 170, 176, 177, 183, 206, 237,
243

_

_Layout.cshtml · 70

{

{action} · 30, 336, 341

~

~ · 68, 69, 70, 72, 81, 83, 145, 150, 160,
161, 163, 164,

169, 178

<

 · 66, 73, 75, 81, 92, 93, 103, 106, 118, 120,

143, 145, 150, 161, 169, 171, 176, 183, 184, 186,

237, 243

4

404 · 33, 38, 45, 46, 136, 137, 138, 140, 297

A

AccountController · 195

AccountModels.cs · 197, 208

action attribute · 177, 179

Action Filters · 25, 48, 49

action methods · 45, 139, 432

action results · 39, 45

Action Results · 25, 39, 49

ActionLink · 57, 65, 66, 69, 72, 73, 75, 76, 78, 81, 92,
124, 137, 138, 140, 141, 143, 145, 150, 161, 174,
221

Active Target Framework · 454, 477

AJAX · 158, 159, 174

AjaxOptions · 170, 177, 183

AntiForgeryToken · 80, 144, 149, 206

ApiController · 348

App_Data · 108

App_Start · 32, 162, 169, 227, 335, 347, 358, 360

appSettings · 294

Arrange · 272, 275

ASP.NET · 26, 31, 68, 69, 160, 195, 232, 237, 247,

272, 275, 296, 297, 335, 340, 412, 413, 421, 438,
439, 448, 451

ASP.Net MVC · 26, 30, 68, 70, 83, 160, 189, 196, 247,
293, 294, 298, 335, 344, 374

ASP.NET routing · 340

ASP.Net vNext · 440, 446, 448, 451

ASP.Net Web API · 344

ASP.NET Web Forms · 421

AspNetLoader.dll · 476

asynchronous · 170

AuthConfig.cs · 227

Authorize · 49, 189, 212, 213, 215, 216, 220, 226, 435

Azure · 8, 293, 294, 324, 330, 333, 421, 424, 439, 441,
442, 468, 469, 470, 471, 472, 473, 474, 483

B

BeginForm · 80, 82, 144, 145, 149, 170, 174, 175,
176, 177, 183, 206

BundleConfig.cs · 429

C

Catch-All Parameter · 337

checkbox helper · 90

click event · 186, 187

client validation · 159

code blocks · 64, 65, 69

code expressions · 64, 65, 69

Code First · 98, 111

code first migrations · 316

config.json · 451

ContentResult · 39

ContentResult, · 6

Controllers · 25, 26, 52, 101, 126, 134, 139, 337

controllers, · 49, 337

CSRF: CSRF · 223, 226, 230

CSS · 43, 386, 388, 413, 428

custom · 32, 49, 65, 72, 107, 108, 122, 157, 162, 169,
179, 186, 188, 200, 204, 207, 234, 237, 249, 294,
295, 301, 333, 377

D

DataAnnotations · 154, 197, 200

DbContext · 100, 108, 197, 200, 201, 269

DbContext, · 100, 269

DbSet · 100, 108, 111, 198, 201, 269

Debugging Routes · 334, 338

Dependency Injection · 343, 356, 362, 363, 372, 439,
440, 441, 461

DropDownList · 88

E

Edit action · 79, 148

ELMAH · 231, 247, 252, 253, 254, 255, 256

EmptyResult · 39

Entity Framework · 96, 97, 122, 124, 141, 200, 304, 461

F

Facebook · 190, 226, 227, 421, 424

Facebook, · 190, 227, 421, 424

field-validation-error · 89

file system · 43, 313, 335, 484

FileContentResult · 39

FilePathResult · 39

FileResult, · 6

FileStreamResult · 39

FilterConfig · 31, 162, 197

filters · 23, 49, 201, 435, 437

Filters · 31, 162, 196, 197

forms · 190, 410

G

Get method · 354

GetVirtualPath · 341

git · 441

H

HomeController · 26, 101, 269, 272, 275

HTML 5 · 97, 162, 175, 377, 413

HTML encoded · 60

HTML encoding · 61

HTML Helpers · 50, 77, 78, 86, 87, 95

HTML property · 86

Http · 45

HTTP · 4, 6, 22, 26, 39, 45, 211, 234, 237, 273, 299,
338, 344, 348, 353, 355, 364, 459

HttpConfiguration · 349

HttpContext · 274

HttpNotFound · 127, 128, 134, 135, 136, 139, 336

I

IBuilder · 459

IgnoreRoute · 335

IIS · 29, 30, 293, 294, 299, 300, 301, 309, 320, 322,
323, 333, 345, 440

Index Action · 213, 268, 436

Index method · 52, 118, 120

Index method, · 120

Index view · 56, 68, 70, 102, 221

J

JavaScript · 5, 22, 156, 158, 160, 175, 176, 177, 179,
374, 413

JavaScriptResult · 39

jquery · 69, 160, 161, 163, 164, 169, 178

JQuery function · 178

JQuery Mobile · 374, 411

Jquery.Validate · 160

json · 451, 452, 453, 460, 463, 483

JSON: json · 39, 44, 159, 344, 349, 350, 355, 363, 439, 440

K

KLR · 440, 484

L

layout property · 70

Linq · 26, 52, 99, 100, 101, 108, 111, 126, 133, 139,
142, 200, 217, 227, 262, 263, 265, 266, 268, 270,
271, 273

LINQ · 55, 79, 88, 94, 96, 97, 114, 116, 119, 122, 141,
142, 166, 266

Log4Net · 247

logins · 276

LogOn · 340

M

MapHttpRoute · 350

MapRoute · 30, 32, 336, 337

membership · 209, 215, 217, 218, 226, 230, 421

Membership · 189, 196, 210, 216, 218, 421

Migrations · 96, 107, 109, 110, 111, 157, 165, 217,
219, 307

Migrations, · 219

Minification · 160

model binding · 83, 154

model state · 89, 152

Models · 56, 58, 65, 72, 73, 75, 80, 91, 92, 99, 100,
101, 102, 104, 106, 108, 111, 112, 117, 120, 126,
134, 137, 139, 140, 142, 143, 144, 149, 166, 169,
171, 176, 183, 184, 195, 197, 200, 201, 206, 217,
227, 261, 262, 263, 265, 266, 269, 270, 271, 360,
413

modernizr · 68, 160, 161, 163

N

NuGet packages · 440, 441

O

OAuth · 189, 190, 226, 227, 230

OAuthWebSecurity · 203, 227

OpenId · 226, 230

OpenID · 189, 190

P

Package Manager Console · 109

packages: Package · 451, 481, 482

parameters, · 32, 179

Partial Views · 50, 91

PartialViewResult · 39

publishing: Publishing · 329

PUT · 343, 370

R

RadioButton · 89, 90

Read/Write actions · 124

readonly · 359, 360

Redirect · 39, 211

RedirectResult, · 6

RedirectToAction · 53, 54, 84, 127, 128, 129, 135,
136, 146, 152, 208, 221, 226

RedirectToLocal · 196, 203

RedirectToRouteResult · 39

RedirectToRouteResult, · 6

RegisterRoutes · 31, 162, 197, 335, 337, 338, 432

roles · 190, 215, 216, 218, 219, 230, 391

Roslyn · 441, 462, 468

Route Constraints · 334, 342

Route definitions · 335

route values · 170, 341

RouteConfig.cs · 32, 335

RouteData · 31, 35, 94, 341

routing engine · 30, 45, 78, 143

S

scaffolding · 5, 34, 52, 56, 91, 95, 133, 136, 149, 419,
431, 438

scripts folder · 159, 160, 177

Search action · 34

Security · 107, 189, 197, 200, 217, 268

seed method · 112, 166, 219, 309

segments · 336

startup class · 425, 459

Startup class · 439, 451, 459

T

TDD · 257, 258, 260, 263, 264

Testing · 257, 292, 294, 440

testing, · 292

textbox helper · 87

TextBoxFor · 207

TryUpdateModel · 83, 84

Twitter · 227, 247, 421, 424

U

URL: url · 30, 32, 35, 37, 38, 39, 40, 41, 42, 78, 79, 83,
177, 179, 195, 210, 328, 333, 335, 336, 338, 340,
341, 342, 349, 351, 355, 368, 377, 473

UrlParameter · 30, 336, 337

UseBrowserLink · 461

users, roles and · 190

V

ValidateAntiForgeryToken · 49, 127, 128, 134, 135, 136, 195, 208, 220, 226

ValidateInput · 49

Validation · 86, 123, 154, 160, 175, 281, 287, 439

ValidationSummary · 50, 77, 80, 86, 144, 149, 206

validation-summary-errors · 87

ValuesController · 348

version · 30, 90, 111, 112, 160, 163, 165, 166, 169,
178, 217, 241, 246, 299, 338, 374, 380, 389, 400,
407, 415, 439, 440, 441, 446, 448, 453, 458, 477,
482, 484

View engines · 51

View State · 2

ViewBag · 26, 27, 31, 57, 60, 64, 65, 68, 72, 73, 75, 80,
88, 92, 94, 95, 101, 102, 106, 117, 120, 137, 140,
143, 144, 149, 160, 169, 176, 183, 206, 272

Viewdata · 50, 94

ViewData · 60, 94, 95

ViewDataDictionary · 94

ViewResult · 6, 39, 272, 276

views · 2, 3, 49, 51, 54, 60, 68, 70, 91, 93, 95, 116,
125, 194, 195, 196, 242, 297, 377, 389, 440

Views folder · 58, 70, 206, 241

Visual Studio · 26, 30, 45, 51, 58, 62, 83, 98, 100, 109,
133, 159, 164, 183, 196, 223, 241, 242, 248, 259,

276, 279, 311, 313, 317, 330, 339, 413, 421, 439,
441, 442, 444, 446, 448, 483

Visual Studio 14 · 439, 442, 444, 446, 448

VM · 439, 441, 442, 443, 444, 445, 468

W

Web API · 343, 344, 345, 346, 348, 351, 352, 353, 357,
362, 363, 372, 415, 421, 438, 439, 440, 448, 461

Web API, · 24, 363, 421

web.cmd · 482

Windows Authentication · 21, 190, 193, 230, 424

X

xhtml · 224

XSS · 95

Y

Yellow Screen of Death · 462

About the Author

Rahul Sahay is a software developer living in Bangalore, India. Rahul has been working in various aspects of the software development life cycle since 7 years, focusing on Microsoft technology-specific development. He has been part

of the development in different applications, ranging from client applications to web services to websites.

Rahul is a Senior Consultant at Capgemini (http://www.capgemini.com/). But he works for Dell R&D, on their premier e-commerce portal("http://www.dell.com/account"). His roles and responsibilities at this project are very tech-oriented like analyzing exiting use cases and taking the new requirements to add features on the existing segment. Prior to Capgemini, he has been associated with Mindtree and TCS. He is also active blogger, his

writings can be viewed at http://myview.rahulnivi.net/. You can also refer his professional profile @ http://in.linkedin.com/in/rahulsahay19. Or follow him at twitter "@rahulsahay19".